A NUMERIC/SYMBOLIC APPROACH TO MACHINE TOOL SUPERVISION

By

TAEHWAN YOON

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1990

ACKNOWLEDGEMENTS

The author would like to thank his supervisory committee for their guidance and assistance in doing this work. In particular, the author wishes to thank Dr. Jose C. Principe for his guidance, support, and encouragement. The author is deeply indebted to Dr. Jack R. Smith who helped make this work possible in all aspects.

The author would like to gratefully acknowledge the collaboration of Dr. Jiri Tlusty, the Machine Tool Laboratory at the University of Florida, for the test data and helpful discussions. He is also grateful to Dr. Shamkant Navathe and Dr. A. Antonio Arroyo for their kindness and helpful comments.

Appreciation is owed to Russel Walters for his editorial help, to Seunghun Park for the utilization of Time Domain Analysis Tool (TDAT), and to Hong Young Ahn, Ju Sung Park, Chongtai Kim, and Haan Go Choi for their encouragement and moral support.

Finally, special recognition is hereby expressed to his wife, Young-Sook, for her constant understanding and encouragement, to his daughter, Jee-In, for her consideration in giving up so many weekends and evenings of family time, and to his parents and his in-laws for their commitment to education and guidance for many years that resulted in this dissertation.

This work was partially supported by grants from the Florida High Technology and Industry Council (1989) and from the National Science Foundation, DDM-8908786.

TABLE OF CONTENTS

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

A NUMERIC/SYMBOLIC APPROACH TO MACHINE TOOL SUPERVISION

By

TAEHWAN YOON

May 1990

Chairman: Dr. Jose C. Principe
Major Department: Electrical Engineering

This work describes the design and evaluation of a sensor-based computer-assisted supervision system, which detects and identifies totally broken, chipped or worn cutters.

A new real-time algorithm (revolution-oriented residual processing algorithm--RORPA) has been developed, which extracts features from x-axis and y-axis displacement sensors attached to the headstock of milling machines. In the raw sensory channel, the algorithm can separate the damage feature from the run-out effect and other reference signals. The RORPA enhances the features of tool breakage, even in the presence of noise, transient and spindle run-out effects.

In addition, a knowledge-based supervision model is developed and applied to enable robust, accurate automated decisions. The model hierarchically integrates the RORPA in a knowledge-based processing environment where rules and objects co-exist. The numeric-symbolic supervision model, which incorporates physical models and empirical knowledge, consists of four tasks: expectation generation, operation monitoring, situation assessment and action execution. The model was developed not only to address the tool

breakage problem, but also to include other, multi-sensor, information as will be required for machine tool supervision.

The two aspects of the supervision model have been implemented in the Texas Instruments Explorer I, a Lisp workstation. In the symbolic environment the object-oriented representations implementing the expectation mechanism and the operation monitoring are written in Flavors, an object-oriented programming facility. The situation assessment uses a rule-based, forward-chaining scheme. The RORPA has been implemented in a multiprocessor board, which consists of four TMS32020 digital signal processors interfaced to the NuBus of the Explorer.

The evaluation of the developed supervision system was performed using a variety of real data cases (194 cases). The overall results indicate that all 80 damaged cases, which included totally broken, chipped or worn cutters, were recognized without a missed-detection. On the other hand, there were only 5 false alarm cases out of the 114 sets of test data from undamaged cutters.

CHAPTER 1
INTRODUCTION

The machine tool industry has a significant impact on the industrial output. Historically it also stands in the forefront of productivity improvement in manufacturing (like numerical control and flexible manufacturing systems). One of the implications of flexible manufacturing and unattended operation is the requirement for automated monitoring. Machine monitoring includes controlling the manufacturing task and identifying possible abnormal occurrences automatically. To implement a fully automated machining environment, several breakthroughs in manufacturing technology are required. In metal cutting, on-line sensing of tool damage detection is one of those tasks.

Statement of the Problem

For unattended machining operation, it is necessary to develop an automated monitoring system which identifies possible abnormalities and initiates protective actions. The possible abnormalities, which we want to detect and identify, include cutter damages and spindle run-out phenomena. Here, we introduce the tool damage detection problem in the milling machine [TI88] and outline the specific requirements of the supervision system.

One of the requirements for an automated monitoring system is to issue alarms for any damage beyond a certain level with a minimum false alarm rate. Therefore, one of the most important tasks affecting the performance of feature detection is how to formulate and set the detection threshold. Setting the optimal threshold for the feature detection of cutter damage and states of wear level in a given machine tool is no small undertaking because the optimal values are rarely constant. A change in the milling conditions (e.g., spindle

speed, feed rate, and depth of cut) or workpiece material can throw off the optimal setting. Consequently, the best value of the threshold for the feature detection of cutter damage at one time is not necessarily the best for optimal processing at another time.

Moreover, the performance of monitoring systems should be insensitive with respect to the signals produced by the transient effect inherent in a milling operation like entering and exiting phases of cutter as well as the signals produced by milling over a slot (i.e., variation in workpiece geometry).

Finally, tool breakage is a condition that can lead to the catastrophic destruction of the machining part and the machine itself, if corrective actions are not taken immediately. Therefore, it is required that the system should detect the abnormalities and issue protective actions very fast (e.g., within a few cutter revolutions) in order to prevent a local damage to spread on the following teeth. In a sensor-based environment we need to collect and process the signal from the sensor in real-time, and make reliable decisions about the machining state.

We summarize the specific requirements of the supervision system for tool damage detection/decision as follows:

1) Reliable and robust detection/decision of damaged tools: it is desirable to maintain perfect detection of tool breakage and minimize false positive errors.
2) Practical sensors: it should be easily located in the machine and be applicable to high speed milling.
3) Fast response: the maximum response time for tool damage decision should be fast enough to prevent a local damage to spread on the following teeth.

Methodology

From the viewpoint of automation, numerical control (NC) of machine tools is still today the state-of-the-art technological development in the factory floor. Indeed it performs very well in the task it was designed for, i.e. to program the sequence of steps and to set the

machine tool parameters. However, NC is a kind of open loop control system, because it does not check for accuracy, chatter, tool wear, and tool breakage during the task. These aspects are decisive for efficient machining, especially in the demanding applications of high-speed, high-power (HSHP) machining and drilling. In these applications a multiplicity of sensors measuring several crucial parameters are required to assess how the task is being executed. Machines have a task schedule (in fact specifications in 4 dimensions, i.e., 3 space coordinates and time), so information gathered from the sensors can be processed and compared to the initial plan. Decisions are then made regarding how well the task is progressing. The problem formulated in this manner belongs to the traditional (analytic) approach.

However, what makes the traditional approach to machine tool monitoring virtually impossible to solve analytically is the huge number of variables in the process and the variety of ways in which errors and alarm conditions can occur. Real life environments are unpredictable, nonlinear and ill-defined, so the present analytic theory is not powerful enough to cope with their diverse nonstationary nature. Presently, there is not a system that can achieve reliable unmanned machine tool supervision. On the other hand, humans do a marvelous job in supervision tasks by combining sensory information with empirical knowledge. A trained operator can sometimes pinpoint a worn-out cutter just by sound, and almost always a broken one in the midst of a very noisy environment.

The supervision of a machine tool requires monitoring several signals associated with the machine status and it must operate continuously. Its activity is driven by periodically sampled data through sensory channel and by occasional requests from operators. In this dissertation we propose a new multi-sensor, hybrid (numeric/symbolic) computer based system that can autonomously supervise machine tools. Specifically, we consider the detection of machine tool damage as a problem domain and use x-axis, y-axis displacement sensors as the sensory channels.

At the present state of technology, a top-down approach to the machine automation problem in cooperation with a bottom-up approach seems to be possible. This method hierarchically integrates real-time signal processing algorithms that run in dedicated digital signal processing (DSP) chips for feature extraction, into a knowledge-based processing environment where rules and objects co-exist. Signal processing will quantify the sensor data and perform the needed data reduction. It will also increase the signal-to-noise ratio without neglecting important features. Without this step the knowledge-based environment would overflow with data, and not be able to react in real-time. Without using the knowledge-based approach where human empirical knowledge can co-exist with physical models, the system could not algorithmically set adaptive thresholds in dynamic operation conditions, formulate hypotheses about faults, and confirm or deny them. These lead to a hybrid approach that combines signal processing algorithms and knowledge-based processing to accomplish unmanned machine tool supervision. Therefore the system can make reliable decisions about the machining state by integrating various sources of information. The hybrid method proposed here, remnant of the way human supervises machining processes, uses the most of the two techniques.

It is important to stress that real-time knowledge-based monitoring requires multiprocessor systems that can concurrently work in a hierarchical configuration. In a changing environment, inputs can not be shut off without dramatic consequences, so while parts of the system are engaged in searching or evaluation, other parts must continue to process valuable input data.

Rationale of Using a Knowledge-Based Approach

Waterman [Wa86] has stated, "When AI scientists use the term knowledge, they mean the information a computer program needs before it can behave intelligently (p. 16)." In the machine tool supervision problem, the "intelligent behavior" means that the

computer will perform the equivalent function of a human supervisor (i.e., it will recognize normal/abnormal conditions and issue typified action).

Knowledge-based systems are computer programs using knowledge and inference procedures for solving problems that are difficult enough normally to require human expertise to arrive at their solution. They structure data and reasoning rules that link the evidence about a problem to derive conclusions. Such systems, which contain the knowledge of a particular expert on a specific subject, may incorporate knowledge acquired from human experts and apply it in novel ways. The internal organization of a knowledge-based system separates the problem domain knowledge from the general strategy for solving problems or methods for interacting with the user. The former collection of domain knowledge is called the knowledge base, while the general problem-solving methodology is called the control module or inference engine. Such a system can grow incrementally as the need arises by expanding the domain knowledge, and can act as an information processing model of problem solving in the given domain. The knowledge-based system can include conventional (procedural) programs, but conventional programs cannot include knowledge-based systems.

Here we describe two major reasons for combining signal processing and knowledge-based approach in machine tool supervision. The first reason is that the conventional detection/decision theory approach cannot solve all the problems under various dynamic operating conditions of machine tools. In most cases of the machine tool operation, although the sensor outputs are sensitive to the level of tool wear or breakage, the variation is dynamic, non-linear and distorted due to the effects of process parameters (like feed rate, depth of cutter, spindle speed and etc.) and noise. The levels of feature components are dependent upon the variation of the workpiece geometry and milling conditions (spindle speed, depth of cut, workpiece material and etc.). The noise component may consist of process noise (e.g. chatter of machine tools) and measuring device noise. These values can be different for different tools and even for different configurations of the

same tool. Their actual values can only be established by experiment. One of the promising solutions is to combine sensory information with empirical or experimental knowledge. This kind of empirical information and knowledge can be encoded more conveniently under the knowledge-based system framework. For example, workpieces with various geometries can be represented in an object-oriented paradigm in order to provide additional information for more robust monitoring, although the signal processing scheme can extract the information about workpiece geometry. Each object can be a workpiece with rectangular shape, one with a slot, one with a convexity, and so on. Each object has distinct properties associated with it and is associated in a network hierarchy that lets it inherit properties of higher-level object (e.g., workpiece, which has a material name and some constants as attributes). These methods provide a natural, efficient way to categorize and structure a taxonomy, such as workpieces or cutting scenarios. Object-oriented representations are unique in that objects can intercommunicate by sending and receiving messages. Objects and message passing provide a natural way for handling processes where the wealth and variety of data configurations determine the reasoning paths.

The second major reason is a need for problem solving capabilities that transcend the limitations (e.g., inconvenient symbolic processing capabilities and implicit knowledge coding in the program) of traditional (numerically oriented) approaches. Even though many elements (e.g., signal processing algorithms and linear approximation model of feature components for rectangular workpiece) of the monitoring tasks can be modeled mathematically or computed using numeric methods, many elements (e.g., noise characteristics, feature interpretation, combination methods of multichannel information and so on) are not sufficiently defined or understood to be amenable to traditional numeric techniques. Although a supervision system could utilize traditional techniques to perform routine tasks, knowledge-based techniques are needed to handle the empirical/ experimental knowledge. As noted previously, even mathematical models are inadequate for many tasks (e.g., cutting with variation in workpiece geometry). In this application, the

traditional (numerically oriented) signal processing environment can be augmented with a knowledge-based approach to incorporate a problem solving scheme, which consists of four tasks (i.e., expectation generation, operation monitoring, situation assessment, and action execution) and has capabilities to cope with dynamic operation conditions, detect abnormalities of the target system, and identify possible inconsistencies between the observed values and the expected values for some important parameters. In this way, computing capabilities more robust than those traditionally available are needed.

Related Research

In this section we will briefly describe previous works in two different areas related to this research: sensing schemes for tool damage detection and real-time knowledge-based systems.

The development of on-line sensing schemes for tool damage detection is an active area of research. A comprehensive review of sensors which monitor the cutting process is provided in [TI83]. The detection of tool breakage in milling has basically followed two approaches: sensors which monitor the cutting process indirectly (i.e. acoustic emission, vibration, sound intensity), or cutting force sensors (i.e. dynamometer, spindle current/torque sensors). One of the approaches is based on the acoustic emission (AE) which is the low amplitude, high frequency stress wave generated due to a rapid release of strain energy. It uses the AE signal produced by the tool breakage [Do80]. Typically the signal has a very broad spectrum (100 KHz), well above the noise produced by the normal machine operation. However, investigators do not agree on the appropriate signal bandwidth [Ka82]. The spectrum shape is also influenced by the cutting parameters, but no clear relationship has been established [Le88]. It is also not clear how to distinguish entry and exit transient bursts from tool breakage. Therefore, feature extraction and threshold setting are difficult to accomplish, even when parametric spectral models are utilized [Li87]. Applications of AE

based techniques in metal cutting are discussed in a number of studies [Iw77, Mo80, Ka81, Li87, Ra87]. The AE signal is obtained from a very high frequency piezoelectric transducer whose frequency range is from 0.1 to 1 MHz. Various processing methods, i.e., both time-domain and spectral analysis, have been used with multi-feature extractions and correlation. In [Li87], cutting forces were also simultaneously measured in order to obtain information for threshold setting. In [Ra87], a neural network is discussed utilizing both cutting force and AE signal. The relation between AE and machining parameters seems unclear.

The other approach is based on the changes in periodicity of the milling force. It uses the well-established relations between cutting force and tool breakage. The milling force can be sensed either directly in the force or torque signal or indirectly from a vibration or sound signal. Several sensors have been utilized, ranging from a dynamometer [Ta87], to monitoring the spindle motor current [Ma82]. It has been shown [Tl83, Al87a] that analysis of the milling process is possible with simple robust processing using the milling force variations. However, the measurement of the milling force is still an impractical problem. The displacement signal has been shown to provide reliable indication of cutter breakage [Tl88]. An extensive study, which proposed a synchronous sampling of the displacement signal and covered different milling speeds, materials, workpiece geometries, and depth of cut, showed that the signals between a damaged and a good cutter can be visually distinguished [Ta88]. This dissertation extends the research and addresses the setting of the thresholds adaptively such that reliable detection can be achieved with automated methods. Towards this goal both theoretical methods using time series analysis [Al87a] and pattern recognition [Is88] have been utilized, but the state-of-the-art is far from being adequate. In this dissertation we use x-axis and y-axis displacement signals as the sensory information.

In addition to the above approaches, some other methods have been proposed: Reference [Ma82] discusses in-process detection of tool breakage in metal cutting by monitoring the current of the spindle motor; In [Ta86], a sound monitoring system is described which can recognize the sound emitted by the machine during operation, using a speech recognition technique.

Now we will describe some applications using real-time knowledge-based systems. A good survey of real time knowledge-based systems and its applications is presented in [La88]. Three broad categories will be discussed here: aerospace, process control and robotic systems.

In aerospace applications, Flight Expert System (FLES) prototype [Al85] is designed to assist an airplane pilot in the task of monitoring, analyzing and diagnosing faults in airplanes. The system is based on the Hearsay blackboard model. Knowledge sources are sensor-interrupt analyzers that alert for faults, construct hypotheses for interrupt causes and place them in the blackboard. When a verification procedure confirms that a component is inoperative, a diagnostic procedure determines the cause of the fault. Liquid Oxygen Expert System (LES) [Sc85] monitors measurements from the launch processing system during loading of the liquid oxygen into the space shuttle. The measurements emanate from a real time process controller. The system uses a frame based model to determine anomalies and initiate troubleshooting procedures. Predictive Monitoring System (PREMON) [Do87] uses an explicit model of a device to perform real-time monitoring. The system has been tested on the partial model of the mirror cooling circuit of the Jet Propulsion Lab space simulator. Spacecraft Control Anomaly Resolution Expert System (SCARES) [Ha86] attempts to automate the diagnosis of anomalies in the altitude control system of an spacecraft with the goal of achieving spacecraft autonomy.

In real-time process control, Fault Analysis Consultant (FALCON) [Ch84] monitors and analyzes alarm signals in a chemical process plant. It consists of five modules, the supervisor, simulator, monitor, fault analyzer and human interface. The real-time module is the monitor. Material Composition Management (MCM) [DA87] consists of the heuristic and analytical process control associated with a chemical processing plant.

In robotic applications the problem addressed is control and guidance of mobile robots [Mc87]. The described system incorporates a knowledge management module, sensor modules and control modules. High level reasoning and planning are provided by

the management module, which observes sensor information generated by sensor control modules, infers the prevailing situation and generates plans to move to a desired state. Unexpected real time occurrences are handled by the lower level real-time control modules, in order to allow time for the management module to catch up with events.

Overview

This section will present a brief overview of the remainder of this dissertation. There are four important parts in this dissertation. The first part is concerned with the front-end feature extraction algorithms, while the second part describes the numeric/symbolic supervision model for machine tools. The third part describes the implementation of the algorithms and model. Finally, the last part describes the system evaluation results using a variety of real test data. Each of these parts is described in a separate chapter.

In this chapter we have discussed the tool damage detection problem and an approach to solve the problem. This chapter has also reviewed the previous works in the related areas. In Chapter 2 we will present a new real-time algorithm (revolution-oriented residual processing algorithm--RORPA) for feature extraction/detection of damages in machine tools. The methods extracts features from the x-axis and y-axis displacement sensors attached to the headstock of milling machines.

Chapter 3 describes the knowledge-based supervision model for machine tools. This chapter will present a numeric/symbolic model to integrate real-time signal processing algorithms in a knowledge-based processing environment where rules and objects co-exist. This model is proposed to incorporate physical models and empirical knowledge. In addition, this chapter will apply the numeric/symbolic model to the tool damage detection problem.

Chapter 4 describes the implementation of the signal processing algorithms (RORPA) for the front-end numeric feature extraction subsystem and the object-oriented programming and rule-based paradigm for the top-level symbolic processing of machine

tool supervision. The first part describes the front-end signal processing algorithms implemented in a multiprocessor system (i.e., Odyssey board). The second part describes the methods of the symbolic processing implemented in the TI Explorer, a Lisp workstation. This part includes the description of the object-oriented representation for the expectation mechanism and operation monitoring. It also describes a rule-based forward-chaining scheme, in which some feature criteria and final decision rules are encoded.

Chapter 5 will discuss the test and evaluation of the developed supervision system with 194 sets of test data. The first part of this chapter describes the data collection set-up. The second part presents the evaluation results of the system that validate the numeric/symbolic model for the tool damage detection problem.

Chapter 6 will present a summary of the main ideas of this dissertation, and suggest directions in which this work can be extended. Appendix A describes a set of Explorer-based software toolkits, which supports specialized TMS320 libraries for digital signal processing applications. The toolkit routines are provided to interface between the Explorer and the Odyssey board to transfer data and commands. Finally, those readers interested in complete sets of the decision and parameter setting rules can read Appendix B.

CHAPTER 2
FRONT-END SIGNAL PROCESSING ALGORITHM

In this chapter a new real-time algorithm (revolution-oriented residual processing algorithm--RORPA) is proposed for automatic extraction/detection of the damage features in machine tool. The method extracts features from the x-axis and y-axis displacement sensors attached to the headstock of milling machines. The RORPA can separate the features of tool damage from the run-out effect signal and other reference signals produced by normal machining. The residual sequence represents changes in system states. The algorithm separates the component corresponding to normal operation from the component that may contain the features of damaged tooth. The mean displacement signals are used as reference information for the detection criterion. The RORPA enhances the features of tool breakage, even in the presence of noise, transient and spindle run-out effects. Results of damage detection/ identification, based on the experimental data, show the RORPA to be very effective and sensitive in identifying operational states of a milling machine.

Introduction

Monitoring is actually the detection and identification of the changes in a system state (e.g., machine tool) during operation, using the signals of one or several sensors and some information available from other sources. Thus we need to study a sensory signal processing method to extract and detect the features.

It has been shown that the cutting force signal offers very good features for sensing milling cutter breakage. In reality, it is difficult to accurately measure the cutting force in milling. There are various ways of measuring the cutting force in milling, either directly

from a table-type dynamometer or a dynamometer built into the spindle bearings, or indirectly evaluating the cutting force from the spindle motor current, torque, vibration or sound. However, the table-type dynamometer distorts the cutting force signal at high frequencies and restricts the working space during milling operations. As an alternative signal, the displacement signal of the spindle has been used to detect tool breakage directly.

Displacement Signal Processing

The displacement signal collected from the headstock carries information about the average force per tooth, and therefore can be used to sense tool breakage. The problem is that the displacement signal amplitude and shape are dependent upon the workpiece material, the speed of milling, the shape of the part, and the radial and axial depth of cut. Therefore simple thresholds in the displacement signal can produce false alarms (i.e. can alert for a breakage condition when the tool is good), and/or missed detections.

It has been shown in [Tl88] that the approach using the displacement signals provides reliable indication of cutter breakage. An extensive study covering different milling speeds, materials, workpieces, and depth of cut shows that visually the signals between a damaged and a good cutter can be distinguished [Ta88]. In [Tl88, Ta88], the displacement signals, which are sampled synchronously with the spindle revolution, are smoothed to create the averaged displacement signal per tooth period, and then a first difference between the consecutive average values is obtained. This difference signal is observed to detect the tool breakage using a threshold. These previous works showed that it is needed to improve the margin between the difference signal levels of the good and damaged cases in the resonant cases and, to some extent, also in the transient and run-out conditions. Relatively severe noise components corresponding to irregularities within one spindle revolution may be added to the real displacement data. This makes it difficult to perform automatic tool breakage detection.

In practical milling, the cutter may travel over holes, slots and other geometric transients on the workpiece, causing variations in the displacement signals. These unavoidable transients must be clearly distinguished from the tool damage which also causes variations in the displacement signals.

A displacement signal corresponding to each tooth period reflects the interaction between a set of teeth and a workpiece in a given operating condition. The displacement signals are periodic with the tooth frequency. Therefore the average displacement signals per tooth period remain constant during steady state milling. If one of the inserts on the milling cutter is broken, the amplitude balance of the signals is upset. The tooth, which follows the damaged one, will have a larger load since it has to remove the extra material left by the broken tooth. The displacement measurements contain the information that is pertinent to machine tool states.

Since the system should detect the tool breakage within a few revolutions to prevent a local breakage to spread to the following tooth, we should consider the maximum response time associated with the signal processing algorithms.

Based on the above motivations, we studied a new real-time signal processing method for the detection of tool breakage using the displacement signal. The overall processing schematic of the front-end feature extraction algorithm is depicted in Figure 2-1.

Revolution-Oriented Residual Processing Approach

The sequences of samples representing the typical displacement signals of the good and damaged cutters are shown in Figure 2-2. The waveform corresponding to one spindle revolution can be divided into resolution elements for each tooth period due to the synchronized sampling used to collect the data. Each resolution element, or "bin" contains mostly information about one tooth displacement. It is evident from this Figure that the displacement signal shows the breakage condition with an abnormal periodic pattern. For

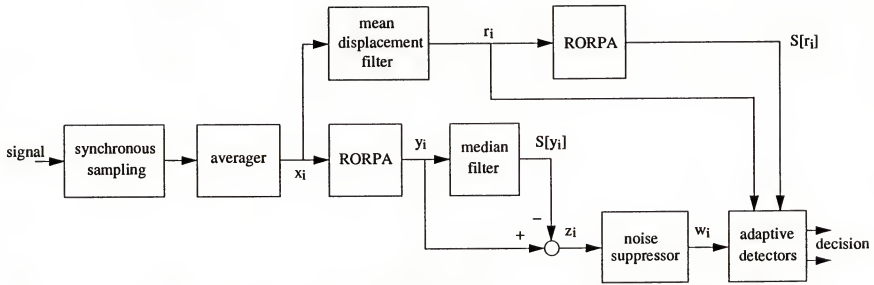

Figure 2-1. The overall schematic diagram of displacement signal processing algorithms using the revolution-oriented residual processing approach (RORPA) for the feature extraction of tool breakage.

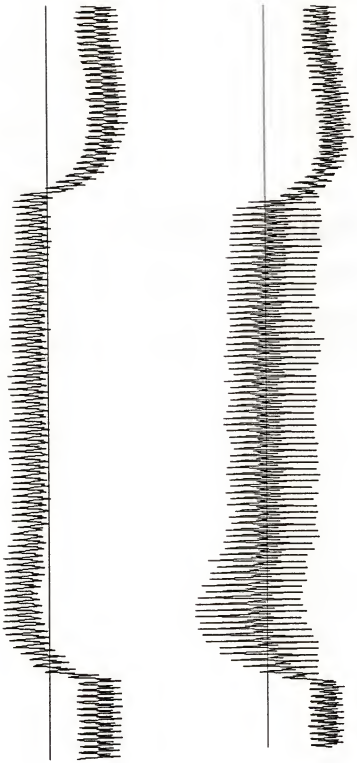

Figure 2-2. The typical displacement signals of the (a) good and (b) damaged cutters.

example, there is a momentary decrease and successive increase in the amplitude of the signals in the bins corresponding to the damaged and next undamaged tooth, respectively. Figure 2-2.a also shows the problem of run-out signal alluded at before. In a perfect setting the ripple seen riding the waveform would not be present. The ripple reflects unbalanced cutting force (uneven force), which can be severe in certain conditions. The fact that the tooth is a contributor to these time dependent variations in the composite signal suggests that time-domain processing techniques should be capable of providing useful detection scheme for such signal features.

We will consider a sample sequence corresponding to one revolution as an ordered set of data, which is processed revolution-wise. By using successive rotation information synchronously for each tooth, we can eliminate the unwanted cyclic phenomena from spindle rotation (e.g., run-out effect) and still observe the changes in a system state.

The rational to use successive rotation information synchronously for each tooth is to enhance tool breakage detectability since it can separate the spindle run-out effect and other signals created by the normal machining. This approach has been called a revolution-oriented processing.

Data sequences created by the RORPA also contain information pertaining to the steady state operation of undamaged machine tools, some undesirable, some useful. In order to eliminate some cyclic components derived from spindle revolution (i.e., run-out effect), it is necessary to extract the difference between the previous and current measurement. The difference sequence, or residual sequence, represents changes in system states.

During the entry and exit transient phases of milling, the displacement signal of the undamaged cutter changes very slowly. A broken tooth produces abrupt signal changes with respect to both the mean displacement value from the previous revolution and the displacement values from adjacent teeth in current revolution. Therefore, the residual sequence can be decomposed into a component corresponding to normal operation and

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}	x_{22}	x_{23}
...	...	...	...	...	...	...	...
...	...	...	x_{i-jN}	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	x_{i-N}	...	...	...	...
...	...	...	x_i	...	...	...	...
...	...	...	x_{i+N}	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	x_{i+jN}	...	...	...	...

Figure 2-3. Illustration of synchronized sample sequence of displacement signals.

another component corresponding to the features of damaged tooth even though they are slightly corrupted by noise. These observations lead to the following signal processing method in which the features of tool breakage can be enhanced, even considering noise, transient and spindle run-out effects.

First, the signal sample is smoothed through averaging, creating an averaged displacement signal $\{x_i\}$ per tooth period. The number of points to be averaged depends on the sampling time and the spindle speed.

Next, for a N-tooth cutter, this data sequence can be partitioned into segments of length N corresponding to one revolution. In continuous operation they can be stacked up vertically, as shown in Figure 2-3. Then for each revolution period (i.e., a point along a row), an average can be obtained to reduce undesired random effects by adding the data x_i's along the vertical line intersecting the data x_{i-jN}, ..., x_{i-N}, x_i, x_{i+N}, ..., x_{i+jN}, as shown in Figure 2-3. This quantity represents a local estimate of the mean displacement per tooth, and its time-varying mean

$$\bar{x}_i = \frac{1}{2W+1} \sum_{j=-w}^{w} x_{i+jN} \quad \text{for } i \geq WN \tag{2-1}$$

is obtained based on a vertical (column-wise) window width of (2W + 1). The window width should be less than the number of revolutions between machine tool start time and workpiece entry time. Otherwise, we can not monitor the system during the early entry phase.

The periodic noise component embedded in the signal is effectively filtered out in the residual or difference signal y_i between the averaged (per tooth period) displacement signal x_i and time-variant mean $\bar{x}_{i-pN}$ at the p^{th} preceding revolution:

$$y_i = x_i - \bar{x}_{i-pN} \qquad \text{for } i \geq (p + W)N \tag{2-2}$$

where p should be greater than W. Both the regular periodic noise and the DC component are removed, and what remain are the component corresponding to the damaged tooth which displays the abnormal pattern, added to the signal corresponding to the transient phases of milling operation which is characterized by a large slow wave. One of the criteria for selecting the parameter p is the desired number of revolutions during which an abrupt event (tool breakage) is remembered in the system. The other is that the parameter p should be greater than the half of the vertical window width. In entry and exit phases, a slow wave due to the transient effect can be amplified by this delay parameter p. Therefore, the above processing produces a new time-dependent sequence which can serve as a representation of the machine tool state even in noisy environments and run-out conditions. This new sequence $\{y_i\}$ can actually be considered an observation space where some features from the first step processing can be extracted.

Decomposition of Residual Signal

The new sequence $\{y_i\}$ still has some noisy fluctuations even under constant milling condition, but their noise level is much decreased. For the damaged tool case, the data

sequence $\{y_i\}$ is composed of the slowly varying part due to geometric transients (entry and exit phases), sharp peaks due to a broken tooth, and a noise component. Thus the sequence $\{y_i\}$ can be considered as

$$y_i = S[y_i] + P[y_i] + N[y_i] \qquad (2\text{-}3)$$

where $S[y_i]$ is the slowly varying part of the data y_i, and $P[y_i]$ is the sharp peaks part of the data y_i, and $N[y_i]$ is the noise part. For the undamaged cutter, $P[y_i]$ will be absent. For the damaged cutter, although there is a noise-like component superimposed on the data sequence, the sequence displays noticeable features (sharp peaks, i.e., double-point doublet-like discontinuities). Such sharp peaks, $P[y_i]$, contain much high-frequency energy, and are essentially indistinguishable from the noisy component, $N[y_i]$, as far as their spectral extent. The slowly varying part $S[y_i]$ can be separated from the sequence $\{y_i\}$ without affecting the component $P[y_i]$. $S[y_i]$ may be used as an information source for the identification of transient phases and adaptive threshold settings in the detection of tool breakage and/or wear level.

The basic concept behind linear filter theory is the separation of signals based on their nonoverlapping frequency contents. Nonlinear filters can separate signals based on whether they are smooth or spike-like. Thus we will use a nonlinear filtering algorithm which can eliminate sharp impulse-like data point in the data sequence and possess a short delay required for real-time applications.

The algorithm based on running medians has the desired capabilities. The properties of running medians are described in [Ra75]. In this application, the size of running medians which are appropriate should be less than two times of the minimum duration of discontinuity which we wish to preserve. It was observed (Figure 2-4) that running medians of 3's and 5's can separate sharp peaks. Although median filtering does not provide sufficient filtering of the broad-band noise-like components, it eliminates sharp peaks in the data sequence.

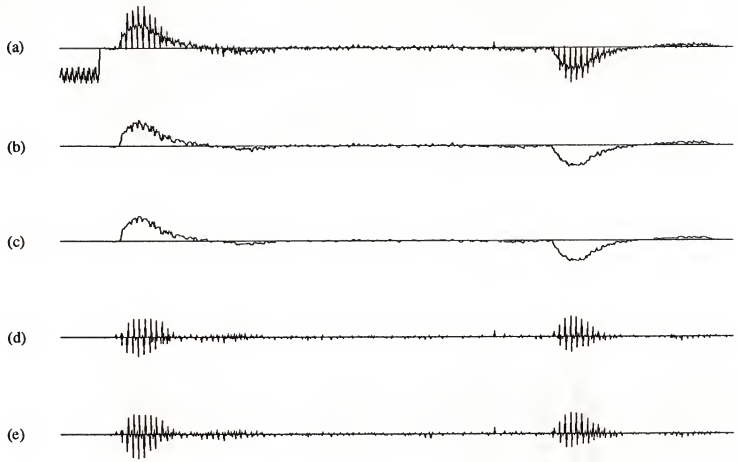

Figure 2-4. (a) A residual sequence $\{y_i\}$ by the RORPA, and the slowly varying component $S[y_i]$ obtained by running medians of (b) 3's and (c) 5's, and the output sequences $\{z_i\}$ corresponding to running medians of (d) 3's and (e) 5's.

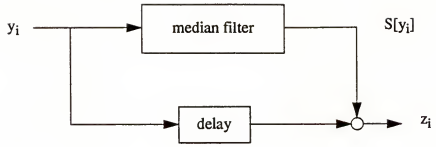

Figure 2-5. A block diagram of the algorithm to separate the doublet-like peaks and slowly varying component from the residual sequence $\{y_i\}$.

In order to extract only the features of the damaged tooth, we will subtract the slowly varying part $S[y_i]$ from the sequence $\{y_i\}$. Figure 2-5 shows a block diagram of the algorithm to eliminate the slowly varying component $S[y_i]$ from the data. The output sequence $\{z_i\}$ may consist of the tool breakage feature $P[y_i]$ and noise component $N[y_i]$. Then z_i is given by

$$\begin{aligned} z_i &= y_i - S[y_i] \\ &= P[y_i] + N[y_i]. \end{aligned} \tag{2-4}$$

Noise Suppressor and Adaptive Detectors

The noise component $N[y_i]$ embedded in z_i may consist of process noise (e.g. chattering of machine tool and so on) and measuring device noise. This value will differ for different tools and workpieces, so the exact value can only be established by experiment. In the current experimental data, it was observed that the levels of the noise component $N[y_i]$ is much smaller than those of feature component $P[y_i]$ and have relatively small variation even though they are dependent on the spindle speed and the transients (i.e. entry and exit) in milling. Therefore, the sequence $\{z_i\}$ is processed to suppress the noise component by the use of a dynamic thresholding method as follows:

$$w_i = z_i \qquad \text{if } |z_i| > z_{th,i}$$
$$= 0 \qquad \text{if } |z_i| \le z_{th,i} \tag{2-5}$$

where w_i is the noise-reduced feature sequence. The dynamic threshold value $z_{th,i}$ can be given by

$$z_{th,i} = C \times \max\{|z_{i+j}| \mid \min z_{i+j} < z_{i+j} < \max z_{i+j} \text{ for } -W_n \le j \le W_n\} \tag{2-6}$$

where C is a constant and the width of moving window is $2W_n+1$ such that $W_n \le N - 2$ where N is the number of teeth on the cutter. The tests with the constant C =1 showed that the noisy fluctuations were sufficiently suppressed while the damage features were preserved. When the noise suppressor constant is C = 1, this scheme sets the current sample to zero unless the absolute value of the sample is greater than the largest absolute value except the maximum and minimum values within the moving window.

Here we discuss how to determine the proper detector threshold levels. The threshold must be set, low enough to safely detect chipping or breakage exceeding a certain desired minimum level. But it also has to be above the level of the signal fluctuations produced by the undamaged cutter in order to minimize false alarm detections.

Detection and false alarm probabilities, on the other hand, are highly interdependent and adjustable via the detection threshold: raising the threshold lowers both probabilities, and vice-versa. Reduction of the detection threshold to improve detectability of tool breakage results in an increase of false alarms. Similarly, an increase in the threshold to reduce false alarms leads to poor detectability of tool breakage. The threshold is typically set by choosing a value based on the perceived trade-off between false alarms and missed detections. Even though we can adopt a fixed threshold based on the noise level for detection of tool breakage, we adopt an adaptive threshold method to improve the performance of detection of tool damage in dynamic operation conditions. The threshold

will be set to be biased towards the false positives, because we will filter them in the knowledge-based processing stage.

It was experimentally observed that for machining started with an already damaged cutter the amplitude level of the signal $P[y_i]$ depends on the slope of the displacement signal of the undamaged cutters during the entry and exit phases, i.e. the level of the slowly varying component $S[y_i]$ in the data sequence $\{y_i\}$. In the case of damage during machining operation, the amplitude level of the signal $P[y_i]$ depends on how fast the damage would spread on one given tooth. In the case of fast spreading damage on a tooth, it is expected that the amplitude level of the signal $P[y_i]$ will be strongly dependent upon the mean displacement signal r_i. As reference information, we may use the slow wave component $S[y_i]$ during the transient phases of machine operation and the on-line estimation value r_i of mean displacement signal during the steady state milling operation.

The relative mean displacement signal with respect to the free milling signal contains useful information for identification of in-process and reference data for the detection of in-process tool breakage. The on-line estimate r_i of the mean displacement signal can be obtained as follows:

$$r_i = \frac{1}{2W_m - 3} \sum_{j=-W_m}^{W_m} x_{i+j} \qquad \text{for } x_{i+j} \in E \tag{2-7}$$

where $E = \{x_{i+j} \mid \min D < x_{i+j} < \max D \text{ for } -W_m \le j \le W_m\}$, $D = \{x_{i+j} \mid \min x_{i+j} < x_{i+j} < \max x_{i+j} \text{ for } -W_m \le j \le W_m\}$ and the window width is $2W_m+1$ such that $1 < W_m < N$ where N is the number of tooth on the cutter. In other word, the mean displacement signal is obtained by averaging the values x_i's, after discarding the extreme values (e.g., two largest peaks and two smallest peaks), within the moving window.

Although the slow wave component $S[y_i]$ can be used as reference information during transient phases of milling operation, it was observed that $S[r_i]$ (see Figure 2-1) contains much smaller noisy fluctuations than $S[y_i]$ while they maintain the same slow wave shapes.

$S[r_i]$, a substitute of $S[y_i]$, is obtained by processing the relative mean displacement signal r_i with the RORPA using the same parameters in the previous RORPA, which is used to extract the residual signal y_i.

Since we can have two cases of tool damages, i.e., in-process tool breakage and starting milling operation with an already damaged cutter, two detectors are used simultaneously. In the detector for in-process tool breakage, the detection threshold is chosen as follows:

$$w_{th,i} = A|r_i| \qquad \text{if } |r_i| > T_{max} \tag{2-8}$$

where r_i is the on-line estimation value of mean displacement signal, and the coefficient A is chosen mainly based on a maximum allowable level of in-process breakage. T_{max} is obtained based on the fluctuation levels of the reference signals. We will set these values, also considering the potential noise and danger caused by small damage which could pass through the tolerance in very shallow cuts. In the other detector focusing on only transient phases, the detection threshold is determined by

$$w_{th,i} = B|S[r_i]| \quad \text{if } |S[r_i]| > T_{max} \tag{2-9}$$

where $S[r_i]$ is a substitute of the slowly varying component $S[y_i]$, and the coefficient B is chosen mainly in consideration of the variation of breakage signals. The latter detector may be used for the detection of the state of tool wear.

Evaluation of the RORPA algorithm

The milling experiments were performed with a workpiece of cast iron on a White-Sundstrand Series 20 Omnimil, a milling machine used for research in the Machine Tool Laboratory at the University of Florida.

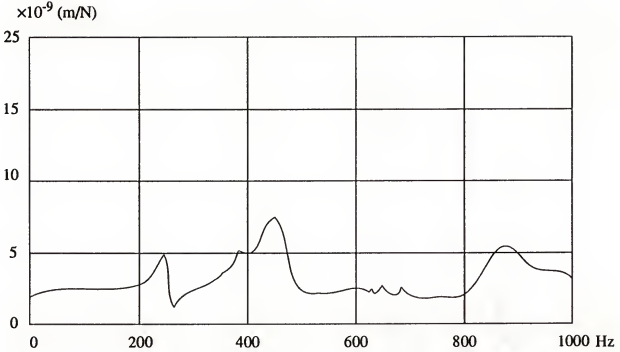

Figure 2-6. Transfer characteristic $T(f_t)$, displacement/force, between the force on the cutter and the relative vibration (displacement) sensed on the spindle.

The RORPA for displacement signal processing, with the window width of 5 samples and the reference time-variant mean $\bar{x}_{i-5N}$ at the 5^{th} preceding revolution (i.e., p = 5), offers extremely good evidence of milling cutter damage. In addition, we used a window width of 5 samples for the median filter, the window width of 15 samples for the mean displacement filter, the window width of 9 samples for the noise suppressor, the coefficient A of 0.1 for the in- process feature detector and the coefficient B of 0.25 for the transient feature detector. This is illustrated using up-milling operation with 1/2 immersion (a/d = 0.5) and with 8 teeth on the cutter. The axial depth of cut b = 0.1 inch (2.5 mm) and feed per tooth, i.e., chip-load c = 0.01 inch (0.25 mm) were used. Now we discuss the experimental results of a machine tool based on a capacitance type displacement transducer located between the housing and the spindle in front of the front bearing.

The transfer characteristic $T(f_t)$ measured between the force on the cutter and the displacement sensed on the spindle is given in Figure 2-6. Within our range of interest it contains two resonant peaks: near 250 Hz and near 490 Hz. The experimental data are selected at 5 different tooth frequencies, i.e., f_t = 80 Hz, 200 Hz, 246 Hz, 258 Hz, and 300 Hz.

The results of test performed at spindle speed ω = 600 rpm, f_t = 80 Hz, are shown in Figure 2-7 for an undamaged cutter and in Figure 2-8 for a cutter with one broken tooth. Each Figure shows the displacement data sequence $\{x_i\}$ on the channel 1, the mean displacement signal r_i on the channel 2, the residual sequence $\{y_i\}$ on the channel 3, the slowly varying component $S[y_i]$ on the channel 4, the feature sequence z_i on the channel 5, the in-process feature detector output w_i/r_i on the channel 6 and the transient feature detector output $w_i/S[y_i]$ on the channel 7.

At the spindle speed ω = 1500 rpm, f_t = 200 Hz, similar results are shown in Figures 2-9 and 2-10 for undamaged and damaged cutter, respectively. In Figures 2-11 and 2-12, in Figures 2-13 and 2-14, and in Figures 2-15 and 2-16, similar results are also seen for the spindle speeds ω = 1845 rpm, 2160 rpm, and 2250 rpm (i.e., f_t = 246 Hz, 258 Hz, and 300 Hz), respectively.

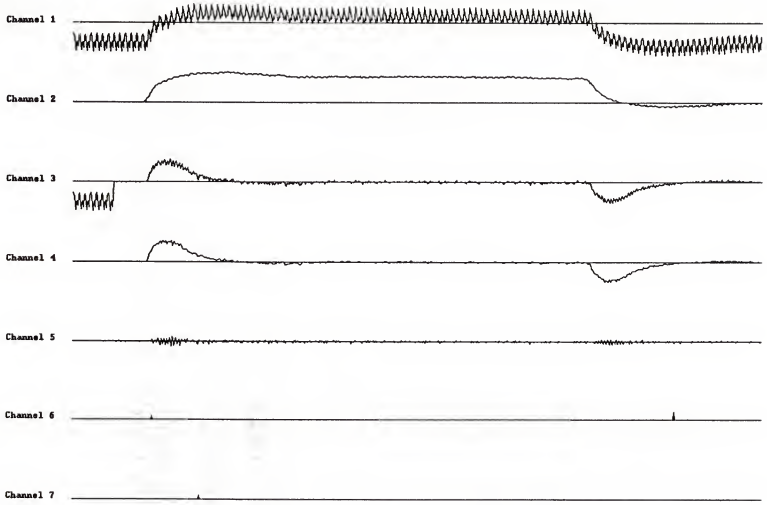

Figure 2-7. The processing/detection results by the RORPA and the detectors for an undamaged cutter at the spindle speed $\omega = 600$ rpm, $f_t = 80$ Hz.

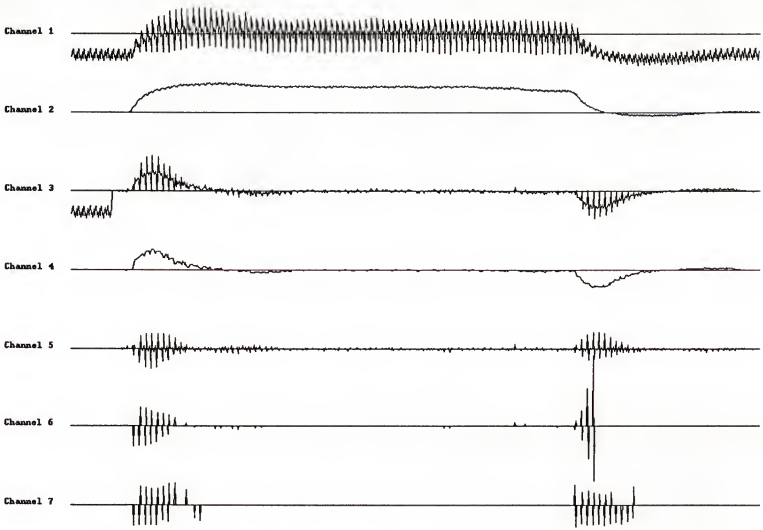

Figure 2-8. The processing/detection results by the RORPA and the detectors for a damaged cutter at the spindle speed $\omega = 600$ rpm, $f_t = 80$ Hz.

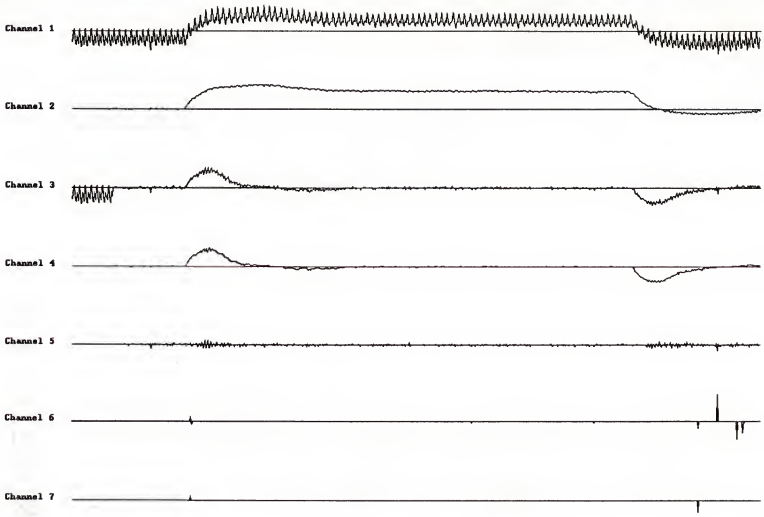

Figure 2-9. The processing/detection results by the RORPA and the detectors for an undamaged cutter at the spindle speed $\omega = 1500$ rpm, $f_t = 200$ Hz.

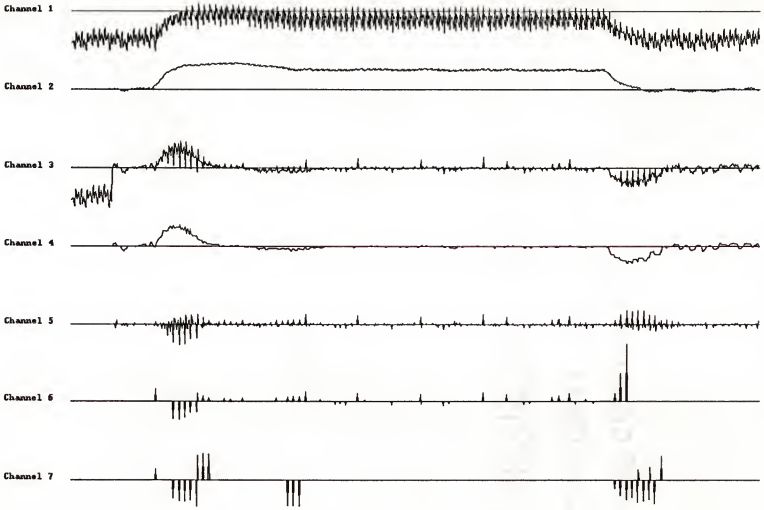

Figure 2-10. The processing/detection results by the RORPA and the detectors for a damaged cutter at the spindle speed $\omega = 1500$ rpm, $f_t = 200$ Hz.

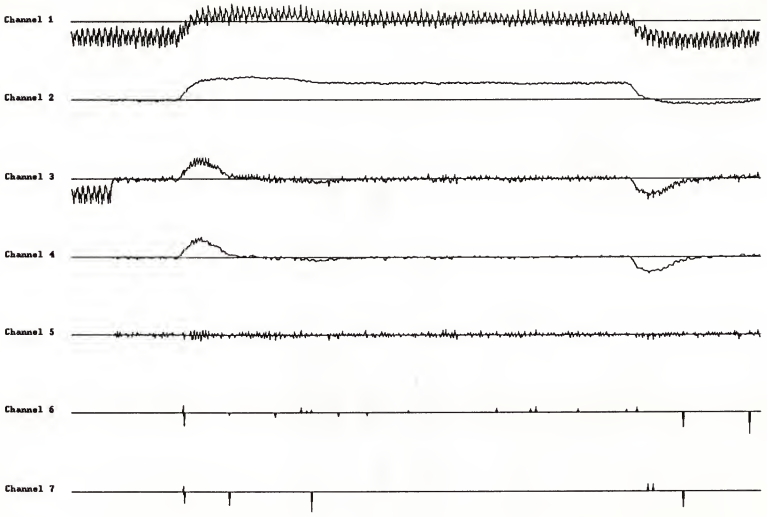

Figure 2-11. The processing/detection results by the RORPA and the detectors for an undamaged cutter at the spindle speed $\omega = 1845$ rpm, $f_t = 246$ Hz.

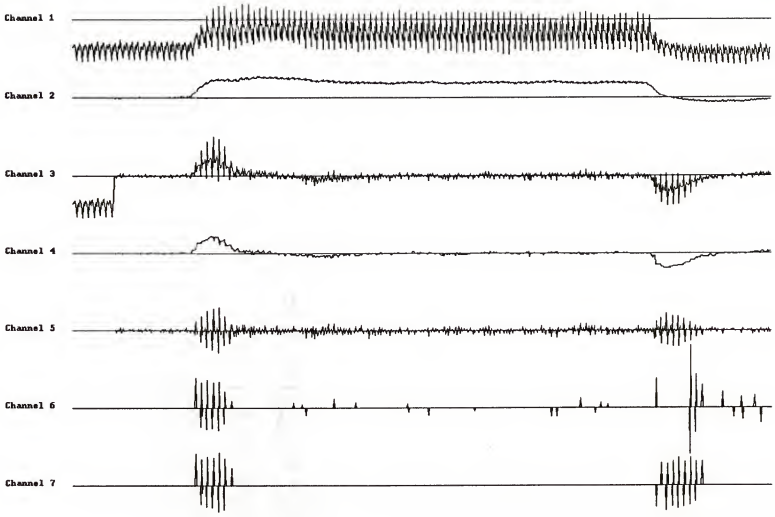

Figure 2-12. The processing/detection results by the RORPA and the detectors for a damaged cutter at the spindle speed $\omega = 1845$ rpm, $f_t = 246$ Hz.

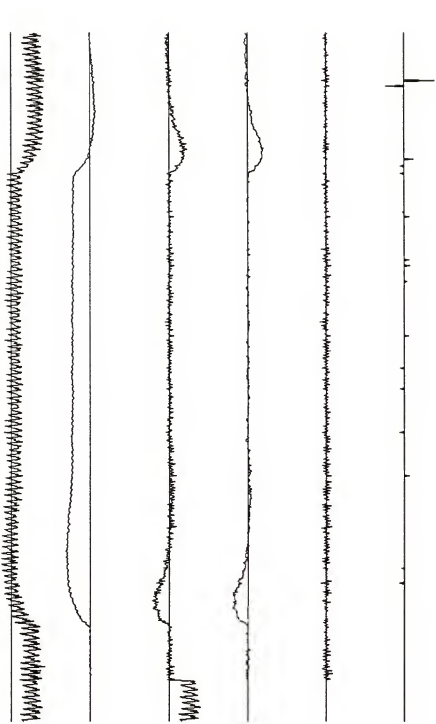

Figure 2-13. The processing/detection results by the RORPA and the detectors for an undamaged cutter at the spindle speed $\omega = 2160$ rpm, $f_t = 258$ Hz.

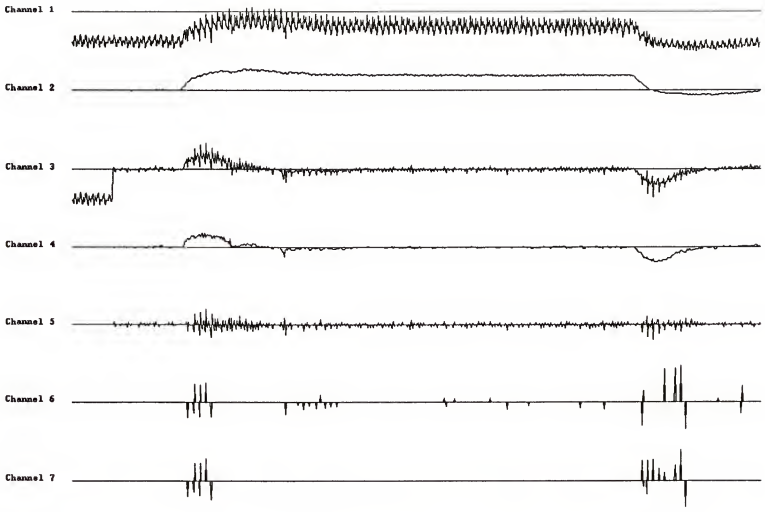

Figure 2-14. The processing/detection results by the RORPA and the detectors for a damaged cutter at the spindle speed $\omega = 2160$ rpm, $f_t = 258$ Hz.

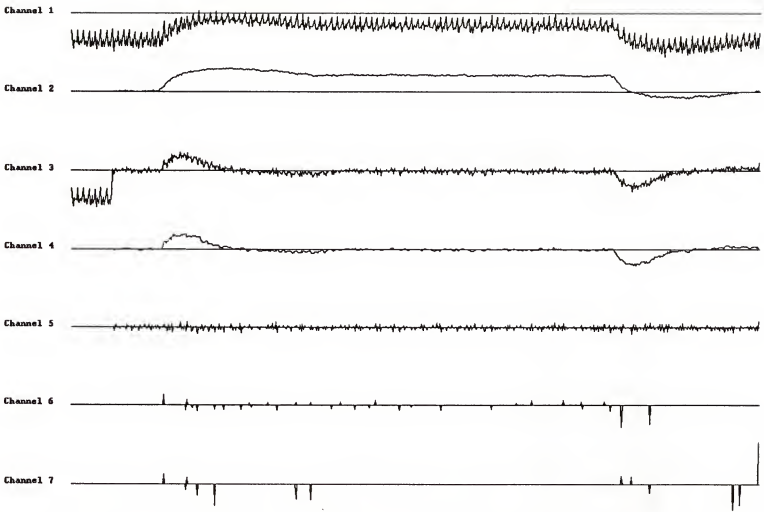

Figure 2-15. The processing/detection results by the RORPA and the detectors for an undamaged cutter at the spindle speed $\omega = 2250$ rpm, $f_t = 300$ Hz.

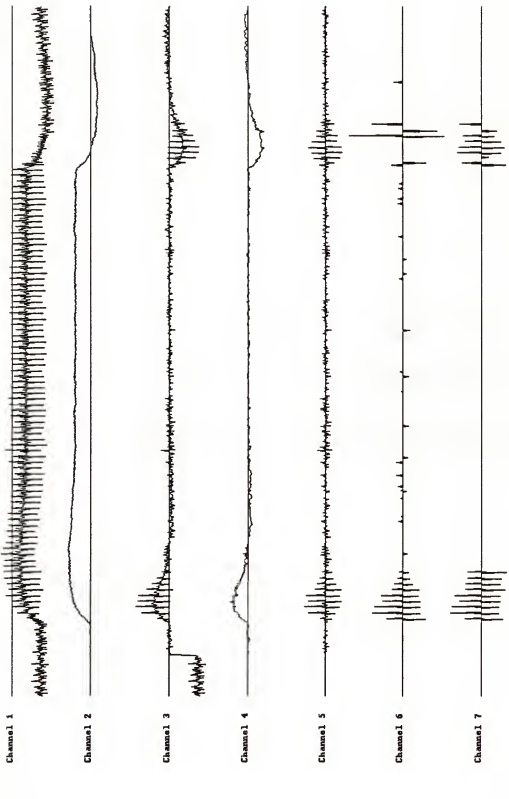

Figure 2-16. The processing/detection results by the RORPA and the detectors for a damaged cutter at the spindle speed $\omega = 2250$ rpm, $f_t = 300$ Hz.

From all the test cases, the abnormal patterns were detected for the broken tooth during the entry and exit transient phases by the interaction between the broken tooth and transient operation of machine tool, even though this is not an abrupt change in the state of machine tool. For the undamaged cutter, it was observed that the detected output signals are sporadic and do not show any periodicity with very small amplitudes.

Discussions

It has been seen that tool breakage in the milling operation causes a momentary decrease and successive increase of the displacement signal, which can be expected from the result of force signal measurement in the case of tool breakage. In the data sequence $\{w_i\}$, one of the abnormal patterns caused by tool breakage is characterized by a momentarily decreasing and increasing pattern, while the patterns by undamaged tool are characterized by runs of only zero's or random sporadic events of small amplitude. This means that the damage features signals have coherent patterns, while the noise is sporadic and incoherent.

The proposed method of displacement signal processing offers the following advantages:

1) It eliminates the severe run-out effect, which is a unwanted component superimposed on the displacement signal.
2) This method to detect tool breakage by the revolution-oriented integration of the noise-reduced sequences $\{w_i\}$ in successive rotations provides increased detectability of tool breakage and reduced false alarm.
3) Its performance for detection of tool breakage can be almost independent of the change of milling conditions and workpiece geometry, i.e. chip-load.
4) It can detect the tool breakage even in the transient phases, i.e. entry and exit phases of milling operation.
5) It can detect the tool breakage near the resonant cases.

In each sensory channel, the decision of tool breakage will be based on the observation of the abnormal patterns during several revolutions. The actual number will be a compromise between maximum allowable response time, detectability of tool breakage, and false alarm. The final decision will be based on the integration of the decisions by x-axis and y-axis sensory channels. This algorithm can easily be implemented with a microcomputer for real-time applications.

Chapter 3 will describe the knowledge-based supervision model for machine tools. This chapter will present a numeric/symbolic model to integrate real-time signal processing algorithms in a knowledge-based processing environment where rules and objects co-exists.

CHAPTER 3
KNOWLEDGE-BASED SUPERVISION MODEL

The knowledge-based supervision system hierarchically integrates real-time signal processing algorithms in a knowledge-based processing environment where rules and objects co-exist. In this research, a numeric/symbolic model is developed which incorporates physical models and empirical knowledge. It is implemented in a multiprocessor architecture. Specifically, we consider the detection of machine tool damage as a problem domain and use x-axis, y-axis displacement sensors as the sensory channels.

Numeric/Symbolic Supervision Model

A knowledge-based system is developed here, which can potentially automate the overall supervision of machine tools. We may regard operation supervision as a combination of signal processing and knowledge-based processing that applies the physical models and empirical knowledge to automatically set various parameters, interpret preprocessed data, assess the situations of the target system, issue appropriate actions, and communicate with operators to give and accept advice. A numeric processing algorithm (RORPA) quantifies the sensor data and processes them for feature extraction. Without this preprocessing stage the symbolic processing environment would overflow with data, and not be able to react in real-time. Signal processing methods alone cannot solve all problems under various dynamic operation conditions of machine tools. The weakness of numeric processing is that it uses the numeric values to reach a decision, which involve a threshold. The data must be numerically processed to see if the threshold is reached or not. Sometimes, there is other type of information such as a priori information, empirical knowledge, or relations between

feature variables that are better captured in a knowledge-based environment. These aspects surface often in practical situations, when the problem has large number of degrees of freedom, and/or unpredictable conditions occur. Machine tool operation is a complex process where the variables that affect machining quality depend on a very large number of conditions and change during machine operation. Therefore, we decided to couple the numeric processing of machining variables with a knowledge-based environment where decisions regarding the conditions of the cutters (i.e. good or broken) could be made.

The overall architecture of this supervision system can be viewed as a two-stage process in terms of processing characteristics: front-end signal processing and knowledge-based processing for decision support (Figure 3-1). The front-end signal processing algorithms are implemented for real-time operation in a multiprocessor system (i.e., Odyssey board). This multiprocessor system contains four digital signal processing modules (TMS32020) and are interfaced in the NuBus of a Lisp workstation (TI Explorer). The knowledge-based processing uses the combination of an object-oriented scheme and a rule-based reasoning, which will be described in detail later.

Front-End Signal Processing

Here front-end signal processing encompasses data acquisition, preliminary processing to accomplish data reduction and feature extraction without substantial loss of information, and basic feature testing. In order to better adapt the signal processing to the specific sensor characteristics, the algorithms should be implemented in local processors. At the same time the arrangement of the local processor functions will keep the data transmission at manageable levels. We use displacement signal sensor output as the sensory channel data.

The detailed methods of displacement signal processing were described in Chapter 2. In this signal processing scheme the displacement signal is converted into a new data

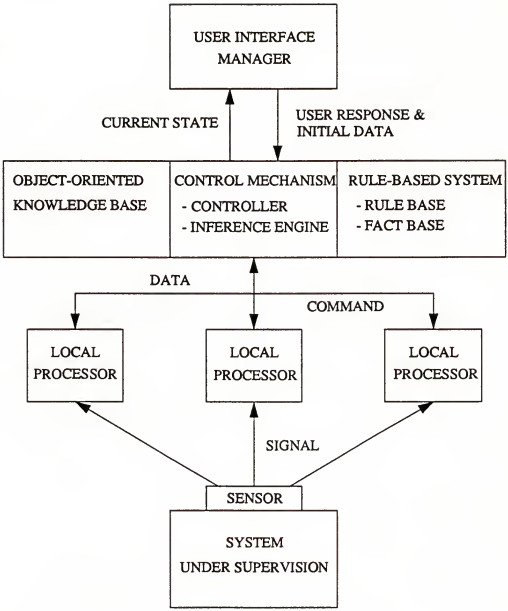

Figure 3-1. Overall architecture of the knowledge-based supervision system.

sequence by the revolution-oriented residual processing algorithm (RORPA), and then the sequence is processed by median filtering to extract the damage feature component. On the other hand, the displacement signal is processed by the mean displacement filter and the RORPA in order to obtain the mean displacement signal and the slow wave signal, each of which reflects the different aspect of machining process. The mean displacement signal and the slow wave signal are used as reference information for the adaptive feature detectors. Since we can have two possible cases of tool damages, i.e., in-process tool breakage and the case where the milling operation starts with an already damaged cutter, two detectors are used simultaneously. The thresholds for the damage feature detectors are dynamically set by the relations in equations (2-8) and (2-9).

The outputs of this signal processing subsystem are sent to the symbolic processing subsystem as two kinds of data: processed signal and detected state. The processed signal outputs include the mean displacement signal, the slow wave signal, the damage feature signal, the feature-to-mean-displacement-signal ratio, and the feature-to-slow-wave-signal ratio for the x-axis and y-axis channels. The detected state outputs are the in-process state and the transient state for both channels.

Knowledge-Based Decision Support

The basic approach in solving the tool damage detection problem is to collect relevant data about the machining process with multi-sensors (i.e., x-axis, y-axis displacement signals), treat this information in order to extract parameters about machining phases, use essential information about the actual characteristics of the task (such as the cutting force from displacement signal). It is possible to combine the feature data received from signal processing subsystem, with the expectations obtained from the a priori knowledge of the task (e.g., workpiece geometry, tool path, spindle speed and etc.) as well as from the physical model. This comparison can not be strict, but it can reduce false positives (i.e., the system recognizes cutter breakage while in reality the cutter is good).

This system is a hybrid (numeric/symbolic) supervision system operating in a dynamic operation environment. It uses knowledge about an approximate milling force model and a priori information of the cutting plan to configure the interpretation process. The core of this processing is the integration of a bottom-up information (the features extracted by the front-end subsystem) and top-down information created by the a priori information of the machining plan and the approximate milling model.

Numeric information from the signal processing part needs to be transformed into symbolic representations. This transformation is performed to allow further processing on the results of the front-end signal processing which may contain false positive symptoms. The false symptoms are sporadic and random, while the damage symptoms show some temporal and/or interchannel relationships. Thus, the top-level knowledge-based processing can reduce the false positives, while maintaining the true positives at a desired performance level.

We may classify monitoring into three basic components: measurements, states, and assessments. At a glance, measurements are the preprocessed sensor data; states are intermediate conclusions of each sensory channel about the operation of a machine tool at a specific time; and assessments are final conclusions about the operation based on multichannel information over an extended period of time. In order to analyze operation supervision, we propose to decompose it into four tasks.

Operation supervision tasks

On-line operation requires a system software architecture (Figure 3-2) that encompasses the generation of expected conditions, operation monitoring, situation assessment, and action execution for more reliable and speedier supervision. The expectation generation task is executed prior to the beginning of machine operation. The other three tasks are performed during machine operation. The expectation generation module receives data from the user, the specification data base or the initial control plan prior to start of machine

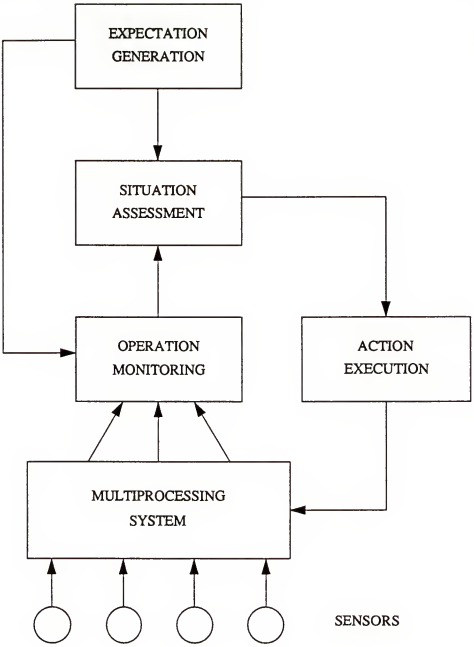

Figure 3-2. System software architecture.

operation. It will not only set several parameters (e.g., the window widths of the RORPA, the threshold for the reference signal detection, and the coefficients for the damage feature detectors) for the signal processing and feature extraction subsystem but also generate nominal values of some variables (e.g., a sequence of milling phases and the confidence level attributed to each sensory channel) from a milling process model and state sequence generator. For example, given the operation conditions and tool path, this expectation module can predict the confidence level of each sensory channel, the abrupt signal change by tool path change and so on. This module can provide reference information for more robust supervision.

The operation monitoring module converts the preprocessed data into a set of symbolic features that describes various aspects of machine tool operation (i.e., transient (entry/exit) or steady state milling phase, and damage symptoms). In this system the monitor of each channel determines the machine tool state at a specific time based on three types of features:

- machining process transient; This is monitored by observing the slow wave component of the data sequence from the front-end signal processing subsystem; This slowly varying component can be used as an information source for adaptive thresholding of tool damage and/or wear level.
- in-process machining operation; In-process state is monitored through the data component which represents the mean displacement r_i with respect to that of free milling and is obtained by averaging the displacement signals, after discarding the extreme values (e.g., 2 largest and 2 smallest values), within the moving window; The mean displacement signals with respect to the signal of free milling can be used as reference information for the detection of cutter damage;
- cutter damage feature; The features of cutter damage can be detected by monitoring the run of consecutive two values of the data component sequence which reflects the state of cutter.

This operation monitoring module compares the states given by the signal processing subsystem with the expected values from the expectation mechanism, and recognizes the operation phase and damage symptoms.

The situation assessment corresponds to the test and validation of the hypothesis regarding machine operations. The system will make final decisions on faulty operation using multi-channel and temporal information. Our approach calls for an extensive and continuous description of the situation in order to formulate a hypothesis. In this way contextual information can be utilized to help diagnose faulty operation.

The action execution has two basic modes: the default mode (operation within normal parameters) and the alarm mode. In the alarm mode, specific tasks that alert the operator and search alternate procedures can be attempted.

Feature representation

At the center of the supervision model is a set of representational features. These features constitute the supervision system's representation of possible states of the target system with which it deals. In the tool damage detection domain, the features might be damage symptoms, milling phases, and outcomes of expectation mechanism.

The representational features are variables that we will assume take on ternary values: when the feature is present, it can take the value +1 or -1 depending on polarity of this feature; the zero value means feature absent. Ternary values possess a tremendous amount of representational power, so it is not a great sacrifice to accept the conceptual and technical simplification they afford. It will turn out to be convenient to denote present with positive polarity, present with negative polarity and absent respectively by +1, -1 and 0, or, equivalently, +, – and 0. Other values could be used if corresponding modifications were made in the following processing schemes. A representational state of the target system is determined by a collection of values, which can be designated by a list of +1's, -1's and 0's.

Symbolic transformation

Figure 3-3 shows the schematic diagram of the integration of the sensory channels and the expectation generation in the top-level knowledge-based processing subsystem. Our approach is to recognize the machining phases and observe the damage features pertinent to each phase. Each milling phase has its own characteristic features.

The feature data used in the symbolic processing environment are the damage-feature-to-mean-displacement-signal ratio, the damage-feature-to-slow-wave-signal ratio, and the detected states (i.e., the in-process state and the transient state) by thresholding the reference signals. The damage symptoms are asserted by thresholding the ratio of damage feature to mean displacement signal and the ratio of damage feature to slow wave signal. The damage feature of the symptoms is represented, using the labels – and + to denote negative polarity and positive polarity, respectively, and no damage is represented as 0. The thresholds have the nominal values of 0.1 and 0.25 for the steady milling phase and the transient milling phase, respectively, but can be updated by the knowledge-based system. The damage symptoms contain time information (pattern). They are represented in the form of a quadruplet list (axis, time-index, feature-polarity, milling-phase). For example it can be asserted as follows:

(x-axis 100 – entry)

The milling phase is represented as an ordered pair of the two state components in the form of (in-process state, transient state). The reference feature components are used to represent the in-process state and the transient state: the mean displacement signal reflects the in-process state; the slow wave signal reflects the transient state. Both states are represented using the labels 1, -1, and 0 to denote the occurrence of the feature with the polarity of the reference signals, or its absence in the corresponding states. All possible representations of the milling phases are described by

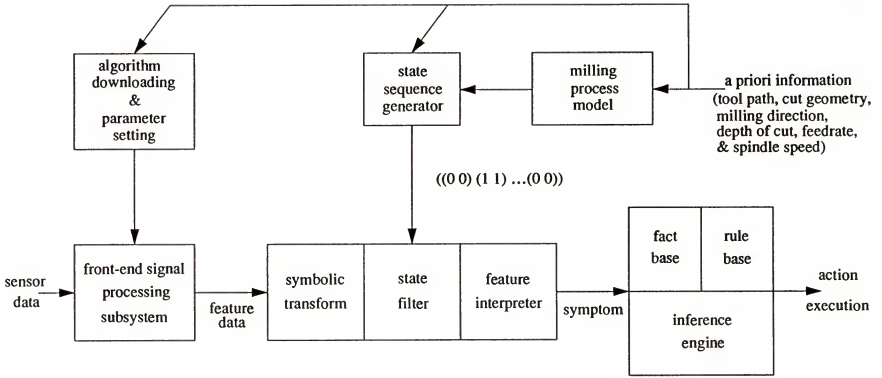

Figure 3-3. Schematic diagram of the integration of the sensory channels and the expectation generation in the top-level knowledge-based processing part.

(0, 0)	=> idle phase
(1, 1), (-1, -1)	=> entry phase
(1, 0), (-1, 0)	=> steady phase
(1, -1), (-1, 1)	=> exit phase
(0, -1), (0, 1)	=> post-exit phase

At a specific time, the feature data from the raw sensory signals can lead to the statements of the form:

state	meaning
(1, 1)	entry phase

These relations represent signal to symbol transformations, and are the primary way sensory signals (bottom-up information) contribute to the classification of milling phases. On the other hand, prior to the machine tool operation, an approximate milling model is used to generate, for the x and y axis, a state sequence of milling phases from the knowledge of the workpiece geometry and cutting parameters. A typical cutting case for rectangle workpiece geometry would be represented in the following way:

$$(0, 0) \rightarrow (1, 1) \rightarrow (1, 0) \rightarrow (1, -1) \rightarrow (0, -1) \rightarrow (0, 0).$$

Milling force model

Milling process mechanics have been extensively studied [Ma41, Ma45, Ko60]. In a milling operation, the actual cutter path is a cycloid. However, the tooth path can be considered as a series of circular arcs with good approximation under the assumption that the feed per tooth, s_t, is much smaller than the cutter radius (Figure 3-4.a). The feed per tooth is

$$s_t = v/(\omega N) \qquad (3\text{-}1)$$

where v is the feedrate, ω is the spindle speed, and N is the number of teeth on the cutter.

In Figure 3-4.b, the mean chip thickness, $h(\theta)$, at the cutter rotation θ may be calculated as

$$h(\theta) = s_t \sin\theta \tag{3-2}$$

The cutting forces are described as a function of the cutting pressure acting on the instantaneous uncut chip area. The instantaneous tangential force, f_T, acting on a single tooth can be expressed as

$$\begin{aligned} f_T &= k_s \times b \times h(\theta) \\ &= k_s \times b \times s_t \sin\theta \end{aligned} \tag{3-3}$$

where k_s is the specific cutting force on the cutting edge and b is the axial depth of cut. The radial component, f_R, of the cutting force is given by

$$\begin{aligned} f_R &= r \times f_T \\ &= r \times k_s \times b \times s_t \sin\theta \end{aligned} \tag{3-4}$$

where r is the ratio of radial to tangential cutting force.

To transform the cutting forces f_T and f_R into the components in the feed and normal-to-the-feed directions, two different methods of generating surfaces by milling operation need to be considered. They are called up milling (Figure 3-4.a) and down milling (Figure 3-4.b). The instantaneous cutting forces, f_f and f_n, in the feed and normal-to-the-feed directions are given by the following relations:

$$f_f = -f_T \times \cos\theta + f_R \times \sin\theta \tag{3-5}$$

$$f_n = -f_T \times \sin\theta - f_R \times \cos\theta \tag{3-6}$$

In multi-teeth milling, there may be more than one tooth active in the cutting. Therefore, the instantaneous cutting forces, $F_f(j)$ and $F_n(j)$, in the feed and normal to the

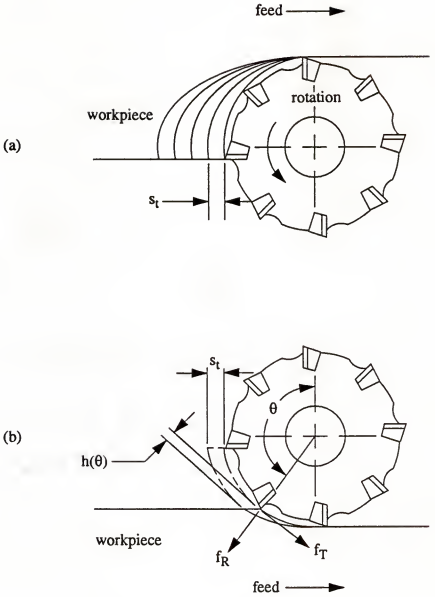

Figure 3-4. Geometry of milling operation and cutting force directions: (a) up milling; (b) down milling.

feed directions at a particular angular position of the N-tooth cutter are the summation of the force components acting on each tooth engaged with the workpiece:

$$F_f(j) = \sum_{i=1}^{N} \delta(\theta_{i,j}) \times f_f(\theta_{i,j}) \tag{3-7}$$

$$F_n(j) = \sum_{i=1}^{N} \delta(\theta_{i,j}) \times f_n(\theta_{i,j}) \tag{3-8}$$

where $\delta(\theta_{i,j}) = 1$ if $\theta_s < \theta_{i,j} < \theta_e$

$= 0$ otherwise

and $\theta_{i,j}$ is the angle of the i^{th} tooth at a sample time j, θ_s is the starting angle of the cut, and θ_e is the exit angle of the cut.

It is known that the instantaneous cutting force is periodic with the tooth frequency. For this tool damage monitoring, we use the information relevant to the average cutting forces per tooth period rather than the instantaneous cutting force values at each sampling time. Therefore, the average forces per tooth period can be written as

$$F_{af} = \sum_{j=1}^{S} F_f(j) \tag{3-9}$$

$$F_{an} = \sum_{j=1}^{S} F_n(j) \tag{3-10}$$

where S is the number of samples per tooth period.

Since the displacement sensors are mounted in the x-axis and y-axis of the spindle housing of the machine tool, the forces need to be represented in the x and y axis of the machine. Then, the average forces in the x and y direction are given by

$$F_{ax} = F_{af} \times \cos\phi - F_{an} \times \sin\phi \quad (3\text{-}11)$$

$$F_{ay} = F_{af} \times \sin\phi + F_{an} \times \cos\phi \quad (3\text{-}12)$$

where ϕ is the angle of feed direction.

Expectation generation

The expectation generation is performed based on the initial control plan prior to the start of machine operation. This module contains the simple milling force model described above, the state sequence generator and the sensory channel confidence classifier. Here we will describe how the operation conditions, workpiece geometry and tool path can predict a sequence of milling phases and the confidence level attributed to each sensory channel.

Since the sensory signals are highly dependent on the workpiece geometry, workpieces with various geometries are represented in an object-oriented paradigm, that will be explained in Chapter 4. Each object can be a workpiece with rectangle shape, one with circular entry, one with circular exit, or etc. Each object has distinct properties associated with it and is associated in a network hierarchy that lets it inherit properties of higher-level objects (e.g., workpiece, which has a material name and some constant as attributes). These methods provide a natural, efficient way to categorize and structure a taxonomy.

The study based on the milling force model and experiments show that the mean displacement signals r_i's in the x-axis and y-axis directions are a function of the radial immersion, the axial depth of cut, the feedrate, the tool path trajectory, and up/down milling. For a given milling operation, we can obtain this a priori information, so the milling process model can generate, directly from the average forces (see equations (3-11) and (3-12)), a predicted value of the mean displacement signal for each segment of the workpiece. As shown in Figure 3-3, this signal is input to the state sequence generator. The state sequence generator calculates the difference of the predicted mean displacement signals between two

consecutive cutting segments. Both these signal predictions can be organized in time as a pair of predicted features that approximately delimits the cutting segments as shown in Figure 3-5. This top-down information is then used to validate the feature data obtained from the front-end signal processing system (see Figure 3-3).

It is shown in [Al87b] that at a certain range of radial immersion the mean displacement signals in one axis are very small, while the mean displacements in the other axis are very large. Provided that radial immersion and other cutting constants are known, we can associate the amplitude value of the predicted mean displacement signal with the confidence level in the segmentation of the milling operation. If this a priori information is not available, we can not use the expectation mechanism and the supervision becomes strictly data driven.

Therefore, we implemented a three-level thresholding operation to establish three different interpretation schemes of the transformed feature data. We classify the confidence

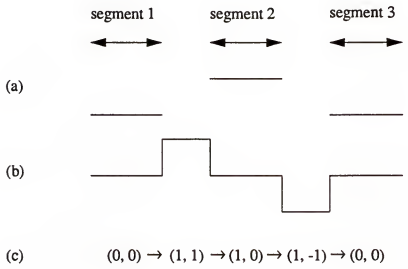

Figure 3-5. The sketch of (a) the predicted mean displacement signals for three segments, (b) the differences of the predicted mean displacement signals between two consecutive cutting segments, and (c) the expected state sequence of milling phases based on the predicted signals of (a) and (b).

level of each channel into three classes: ignorant, ambiguous, and certain. This confidence level is obtained by applying two threshold values (i.e., the state threshold and the ambiguity threshold) to the predicted mean displacement signal from the milling process model. The rules for providing these threshold values, which depend on the workpiece material, are given in Appendix B. When the magnitude of the predicted mean displacement signal is less than the state threshold value, the confidence level is classified as ignorant. In this case, the features from the sensory channel are ignored even though the extracted features show a damage symptom. When the magnitude of the predicted mean displacement signal is greater than or equal to the state threshold value and less than an ambiguity threshold, the confidence level is classified as ambiguous. In this case, we do not use the expected state sequence to interpret the feature data from the front-end signal processing system. This corresponds to a strict data-driven processing. The state detector works only using the milling phases measured from the sensory channel. When the magnitude of the predicted mean displacement signal is greater than or equal to the ambiguity threshold, the confidence level is classified as certain. In this case, the feature interpreter uses the state filter during the feature interpretation. The state filter performs in such a way that the measured milling phase is modified based on a predicted sequence of milling phases generated by the expectation mechanism. For each axis, the feature interpreter and state filter are configured, depending on its confidence level.

Feature interpreter

As we have described in Chapter 2, we can have two possible cases of tool damages, i.e., in-process tool breakage and the case where the milling operation starts with an already damaged cutter. Two kinds of damage symptoms, the slow wave and in-process damage symptoms, are observed for each sensory channel. It was discussed in Chapter 2 that the reference feature signals (i.e., the mean displacement and the slow wave signals) provide sensitive information about the current state of the cutting process.

Thus, damage symptoms are defined as the ratio of the damage feature to reference signal to represent the damage hypothesis for the final decision. The feature interpreter observes the damage symptoms pertinent to either the transient or steady-state phase, as recognized by the state detector/filter. The slow wave damage symptom is observed during the transient phases (i.e., entry and exit phases), and the in-process damage symptom is observed during the steady state milling phase.

It was shown in Chapter 2 that one of the abnormal patterns caused by tool breakage is a momentary decreasing followed by an increasing pattern in the damage feature sequence, provided that the reference feature signals are positive. The level and polarity of the damage features are dependent on the milling parameter translated by the reference feature signal. The amplitude ratio of the damage feature to the reference feature signal shows a good measure of cutter breakage under the varying cutting conditions of our data set. In cases with tool damage, it was observed that a damage symptom with negative polarity, called primary damage symptom, almost always appears. On the other hand, a damage symptom with positive polarity, called auxiliary damage symptom, appears following a primary damage symptom. This means that the auxiliary symptom provides supporting evidence about cutter damage. Thus the auxiliary damage symptom is asserted into the situation assessment module only when it consecutively follows a primary damage symptom.

For example, the milling phase sequence of each channel can be predicted as idle, entry, steady, exit, post-exit and idle phase in order, or as only the idle phase, depending on the operation conditions and other factors. When the a priori information for the expectation generation is not available, the expectation mechanism is not used and the confidence levels of the sensory channels are classified as ambiguous. Suppose that the tool path includes a corner slotting. The displacement signals change abruptly near the corner due to the sudden changes of immersion at the corner. It is possible to predict the corner slotting time from the tool path by a method similar to the above one. However the expectation function to predict the event time is not included in the current system implementation.

Decision strategy

The final decision of tool damage is made on the situation assessment block based on the temporal patterns of the damage symptoms in the x-axis and y-axis channels and the interchannel relationship. The time to reach a decision is dependent on the patterns of the damage symptoms. The decision rules are written in a rule-based programming technique, which will be described in Chapter 4, and are listed in Appendix B.

The decision knowledge base is written around the concept of a three layered panel: symptom, intermediate, and decision panels (see Figure 3-6). The symptom panel is the first level of decision where sets of rules use two symptoms to produce an intermediate hypothesis for the next decision level. The intermediate panel is the second level, where the symptom and intermediate hypothesis can be combined to generate a more probable damage hypothesis for the highest level. The decision panel is the third level, where a final conclusion about tool damage can be made through a combination of a hypothesis and a symptom or with two hypotheses. An updated hypothesis can also be obtained on the decision panel.

This system solves the cutter breakage detection problems by selectively combining symptoms into intermediate hypotheses in order to form further intermediate hypotheses that explain more symptoms within larger time scopes (such as extended periods of observation time) and/or more confidence (such as multiple sources of evidence). This combination process uses domain-specific knowledge to form consistent interpretation hypotheses and to eliminate those which are inconsistent. The domain-specific knowledge sources are based on temporal and interchannel relationships as criteria for combining damage symptoms. Multiple sources of evidence for an event may increase the certainty of the event. Here, multiple sensory channels (i.e., x-axis and y-axis displacement sensors) can provide mutually supporting evidence about a machine tool's state. However, the evidence from each channel may depend on the confidence level of the channel. Thus, this

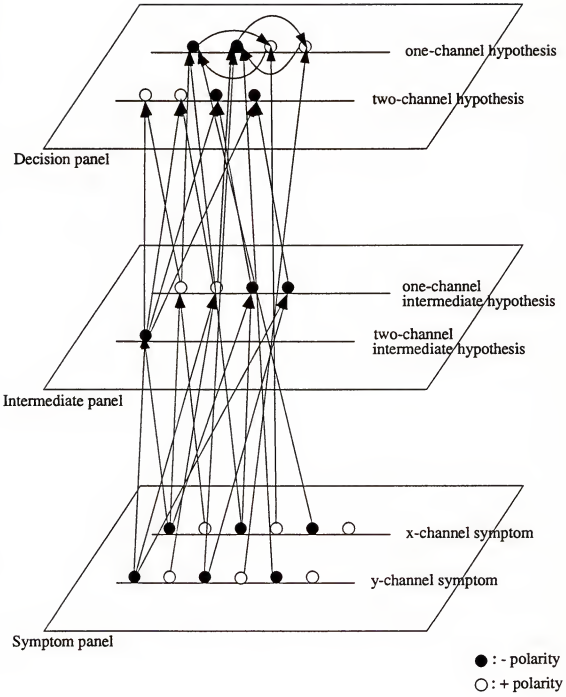

Figure 3-6. The decision knowledge base panels.

decision process can be viewed as a constraint-aggregation process that involves the creation of more encompassing hypotheses and the elimination of unlikely hypotheses as a result of an attempt to incorporate additional observations.

To combine corroborating evidences, we use three kinds of symptom combination rules: primary symptoms' temporal coherency (PSTC), primary symptoms' interchannel synchronism (PSIS), and auxiliary symptom's dependency on the primary symptom (ASDP). The primary symptoms' temporal coherency means that the current primary symptom can upgrade, in the hierarchy of the panels, a previous primary symptom in the same channel when the temporal interval between the two symptoms is a revolution period. The primary symptoms' interchannel synchronism means that the current primary symptom of a channel can upgrade the primary symptom of the other channel when the two symptoms are synchronous events. The auxiliary symptom's dependency on the primary symptom means that the current auxiliary symptom can upgrade a primary symptom in the same channel only when it follows the primary symptom at the next time of tooth period. On the symptom panel, these rule are directly applied. On the intermediate and decision panels, the same kinds of rules are applied between the last primary symptom part of the hypothesis and the current damage symptom. In addition, the above symptom combination rules can be applied only when all the symptoms appear during the same milling phase.

When a newly asserted symptom does not satisfy any of the above symptom combination rules, evidence elimination rules are applied to eliminate the symptoms and hypotheses obtained in the previous revolution.

This situation assessment module applies its knowledge sources to extend and refine intermediate hypotheses so that they form a conclusion meeting the decision criteria. On the decision panel, we use two different criteria in terms of the number of symptoms for the final decision of tool damage: four-symptom and six-symptom rules. The four-symptom rules are usually applied as the default strategy to improve the decision time. These are based on the hypotheses that consist of four damage symptoms. These damage symptoms

should be governed by the symptom combination rules. The six-symptom rules are applied depending on the situation. One of these situations is when the damage hypothesis is formed only in one channel and the milling phase of the other channel is not 'idle'. In this case, a more stringent rule is used, which requires a pattern of three consecutive perfect (both primary and auxiliary) damage symptoms. During the system test phase, we included another six-symptom rule. This rule is applied when both channels assert hypotheses that consist of three damage symptoms but the temporal patterns between two channels are shifted by one tooth period.

Alarm processing

A strategy to address alarms is required. Our approach is to act swiftly, even if not very accurately, provided that the situation can be re-examined later with more data. Towards this goal we created two basic modes of operation, alarm and normal modes. Alarm mode has the highest priority. It can be invoked with preliminary evidence of a failure from a single sensory channel with a reasonably low threshold (i.e., false positive may occur). The system immediately enters a mode that prevents catastrophic failures. However procedures to gather more information about the condition are an integral part of the alarm handling. The system will be able to resume normal operation if during the more extensive system check involving multi-sensor information and likelihood of failure, the condition has been proven to be a false alarm. Alarm actions are controlled by alarm knowledge sources, which we call specialists. Therefore when an alarm is initiated the system already has indication of what may be going wrong, i.e. the specialist automatically creates a focus of attention for the symbolic environment.

An example from the machine tool domain illustrates well this paradigm. Assume that one sensory channel detects a displacement level that can be associated with tool breakage. Tool breakage is the most costly operating condition and should be addressed immediately. To prevent further damage to the tool and machine it is enough to slow the feedrate

of the workpiece. The question now is whether it is a true broken tooth or just an artifact produced by the geometry of the workpiece. To answer this question, more data must be gathered from the same sensory channel and integrated with data from the other sensory channel to check if the whole picture corroborates tool breakage. Notice that this information gathering procedure may be time consuming, but the machine was already placed in a default mode that guarantees no further damage. Machining a few seconds with a slower feedrate is a small price to be paid if one considers that damages to the machine tool can be prevented.

Discussions

One of the main goals of this research is to develop of an information processing model for machine tool supervision. Machine tool supervision consists of four tasks (i.e. expectation generation, operation monitoring, situation assessment and action execution). This can create an accurate decision environment, and the combination of signal processing and knowledge-based decision is able to increase the robustness of the decision.

The model has been evaluated using a variety of real data. For example, the clear relationship between the radial immersion and the mean displacement signals in the x-axis and y-axis directions should be studied since the radial immersion can be a valuable information for feature detection under various geometry conditions.

This is a general model for knowledge-based monitoring. It was developed not only to address the tool breakage problem, but also to be modularly extended to include multi-sensor information as will be required for the machine tool domain.

Chapter 4 will describe the implementation of the supervision model: the signal processing algorithms (RORPA) in a multiprocessor board and the knowledge-based processing in a Lisp workstation. The top-level symbolic processing part is implemented using the combination of an object-oriented scheme and a rule-based paradigm.

CHAPTER 4
IMPLEMENTATION IN A MULTIPROCESSOR ENVIRONMENT

This chapter describes the implementation of the signal processing algorithms for the front-end numeric feature extraction subsystem and the object-oriented programming and rule-based paradigm for the top-level symbolic processing of machine tool supervision.

Multiprocessor System for Digital Signal Processing

The front-end signal processing algorithms are implemented for real-time operation in a multiprocessor system (i.e., Odyssey board), although the system currently runs in a file mode. The Odyssey DSP (Digital Signal Processing) system provides a rapid prototyping environment for the development of signal processing systems by integrating symbolic and signal processing. This multiprocessor system contains four digital signal processing modules (TMS32020) and is interfaced in the NuBus of a Lisp workstation (Texas Instruments Explorer I).

A single Odyssey system consists of four independent processor modules connected by the Odyssey internal bus, called a Signal Processing (SP) bus. Each module contains a TMS32020, 128 Kbytes of data memory, and 16 Kbytes of program memory. The TMS32020 operates with a 5 MHz instruction cycle and can execute at a 10 MIPS rate.

Each DSP module and the Explorer host interface are linked with the SP bus, which has 24 bits of address and 16 bits of data. This SP bus can also be extended off-board to tie multiple Odyssey boards into a large extended memory. All memory, command, and interrupt functions are memory-mapped across all boards. Interboard accesses do not require

host system bus cycles except when accessed by the Lisp processor. Any processor can access any other processor's memory as an extension of its own data memory.

Overview of Toolkit

A set of Explorer-based software toolkits supports specialized TMS320 libraries for math, DSP, and other applications [Ma88]. This toolkit can be thought of as a mini-operating system which allows the application developer to allocate Odyssey resources (processors and data memory) and control the execution of the tasks on these resources through calls to a set of provided runtime Lisp routines on the Explorer. A set of object libraries is provided in the toolkit to allow the application developer to perform a large range of tasks on the Odyssey board, such as data and commands transfer between the Explorer and the Odyssey board.

Corresponding to each set of source and object libraries will be a library control file. The library control will contain interface information about each function in the library. This information will include the number and size of the input buffers, the number and size of the output buffers, the amount of working space, and the number of parameters required to execute the routine. Default values will be included in the library control file for all parameters.

In general, TMS32020 functions in the library will be required to get input parameters, input/output buffer addresses, and working space pointers from a communication block. The application developer is required to divide the task into load modules and specify in which processor each load module will execute. The support routines will allocate data memory in the processors in the accordance with the developer directives.

The Lisp support routines can be divided into two categories, runtime and non-runtime routines, and are described in Appendix A. The runtime routines include routines to allocate buffers, define tasks, download modules, put data to the buffer, get data from the

buffer, start task execution, and stop task execution. The non-runtime routines include the routines to interactively build the libraries and control files.

Implementation of RORPA Algorithms

The signal processing (RORPA) algorithms are partitioned and written into the seven routines: averager, mean displacement filter, slow wave extractor, residual value extractor, damage feature extractor, noise suppressor, and detector routines.

Configuration

Figure 4.1 shows the configuration and program flow diagram of the front-end signal processing routines. This configuration shows a master processor "3" receiving x-axis and y-axis input data in two separate buffers, called inbuf1 and inbuf2. An averager routine which is executed twice in sequence receives these data, and produces two output buffers, called avgout1 and avgout2, containing the two averaged signals. Each of these signals is sent to its own channel processor. Each channel processor has the algorithms for extraction of both the reference signals and the damage feature component. For the extraction of the reference signals, the mean displacement filter routine generates a buffer, called mdfout1 (or mdfout2), for the slow wave extractor routine. This routine produces two outputs. One is a copy of the input, and the other is the slow wave signal. Both are stored in a buffer, called slowout1 (or slowout2), which is used as input for the detector routine. For the extraction of the damage feature component, the residual value extractor routine generates a buffer, called rorpaout1 (or rorpaout2), for the damage feature extractor routine which generates featout1 (or featout2) for the noise suppressor routine. The noise suppressor routine generates noise-reduced damage feature for a buffer, called nsupout1 (or nsupout2), which is used as input for the detector routine. Thus each channel processor generates two buffers for the detector routine. For both channels, the detector routine

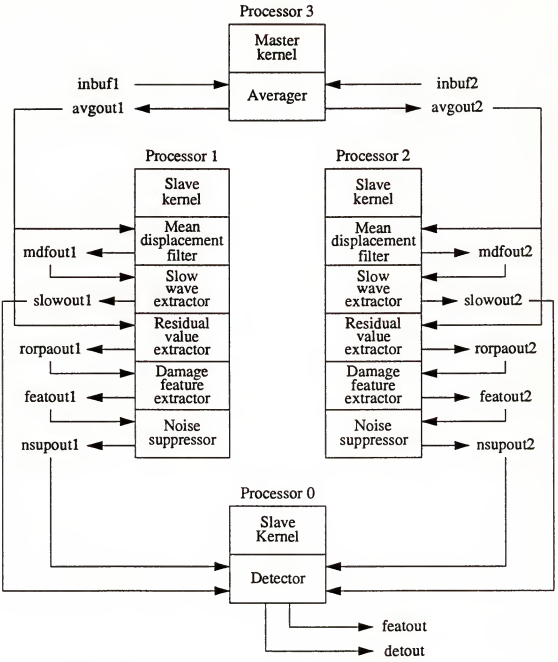

Figure 4-1. The configuration and flow diagram of the front-end signal processing routines.

provides the ratios of the damage feature to the reference signals and two states (i.e., the in-process and transient states) for a pair of output buffers, called featout and detout, which are read by the symbolic processor (TI Explorer).

Parameters for the routines

The seven routines for the RORPA algorithms use the fourteen parameters: NTEETH, SPT, MAWIN, SKIPN, MLWIN, RWIN, DIFDL, GAP, MDWIN, NSWIN, SRATIO, DRDEN, SRDEN, and STH. Now we discuss what the parameters mean and how they are set.

The experimental results of the RORPA algorithms were discussed in Chapter 2. From the experimental test, the RORPA with the revolution-oriented window width (MAWIN) of 5 for the reference time-variant mean at the 5^{th} preceding revolution (i.e., p = 5) offers extremely good evidence of cutter damage.

For notational convenience, we will write

[x] = the greatest integer less than or equal to x.

The parameter NTEETH is the number of teeth on the cutter. This parameter is always known prior to the operation monitoring. The parameter SPT means samples per tooth period. Since the signal is synchronously sampled by a 72-tooth steel gear mounted on the spindle, this parameter is set by the following relation:

$$SPT = 72/NTEETH \tag{4-1}$$

The parameter MLWIN, which is the width of the moving window for the mean displacement filter routine, is set by

$$MLWIN = NTEETH*SKIPN - 1 \tag{4-2}$$

where SKIPN is half of the number discarded as extreme values within the moving window for the mean displacement filter (SKIPN = 2 as the default value). The parameter RWIN, which is the width of the RORPA window, is given by

$$RWIN = NTEETH*(p + [MAWIN/2]) + 1 \qquad (4\text{-}3)$$

The parameter GAP, which is the gap between current data and upper end of window for its reference time-variant mean, is given by

$$GAP = NTEETH*(p - [MAWIN/2] - 1) \qquad (4\text{-}4)$$

The parameter MDWIN is the window width of median filter, which is used in the damage feature extractor. It is shown in Chapter 2 that running median with MDWIN = 5, which is used in this application, can separate sharp peaks.

The parameter NSWIN, which is the width of moving window in the noise suppressor, is set by

$$NSWIN = 2*(NTEETH - 2) + 1 \qquad (4\text{-}5)$$

The parameter SRATIO, which is unity as the default value, is a constant such that the threshold is set as a product of SRATIO and the absolute value of the largest sample after omitting the largest and smallest values within the window of the noise suppressor.

Since we extract the damage feature component and reference components by two parallel paths from a sequence of input data, we should consider the difference of processing delays between the two paths in order to synchronize the damage feature component with its reference components. The parameter DIFDL, which is the difference in delays between half windows of the two paths, is given by

$$DIFDL = [MLWIN/2] - ([MDWIN/2] + [NSWIN/2]) \qquad (4\text{-}6)$$

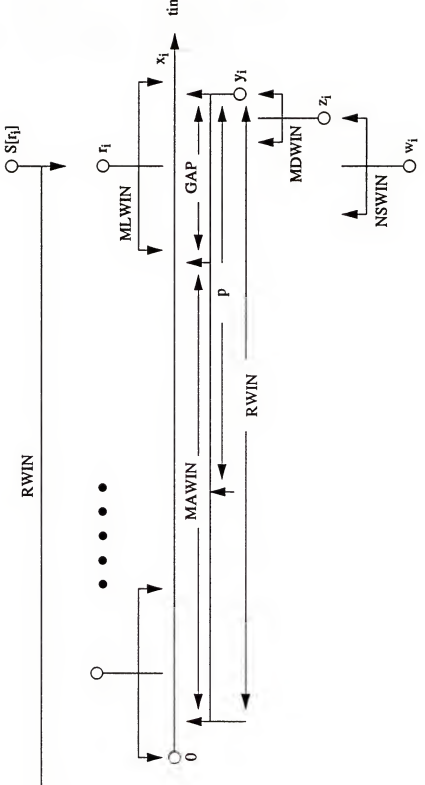

Figure 4-2. The timing relationships of the windows in the signal processing routines.

The parameter STH, which is the threshold value for the state detection (i.e., the in-process and transient states), is obtained based on the noisy fluctuations of the reference signals during the idle milling phase. The parameter DRDEN is a coefficient for the threshold in the in-process tool breakage detector. This coefficient was selected based on a maximum allowable level of in-process breakage, the variation of breakage signals, and the noisy fluctuations of undamaged signal during the in-process milling phase. The parameter SRDEN is a coefficient for the threshold in the detector focusing on only transient phases. This coefficient was selected based the potential noise and danger caused by small damage which could pass through the tolerance in very shallow cuts during the transient phase. The two parameters, DRDEN and SRDEN, can be updated by the parameter setting rules, which is listed in Appendix B, in the knowledge-based system. The threshold parameters mentioned above were empirically obtained.

The timing relationships of the windows in the signal processing routines are shown in Figure 4-2. The windows will be positioned in time so that the noise-reduced damage feature w_i is synchronized with the mean displacement signal r_i and the slow wave signal $S[r_i]$. Since the three functions are sequentially processed in the damage feature extraction path (i.e., bottom part of the Figure), the windows (i.e., RWIN MDWIN, and NSWIN) are positioned considering the processing delays.

Feature extraction routines

Averager routine. This routine receives the signal samples from the input buffer, and then creates an averaged displacement signal x_i per tooth period. This function generates an output buffer for each channel, which is used as input to the mean displacement filter and the residual value extractor routine. It needs one parameter called SPT, which is the number of samples per tooth period.

Mean displacement filter routine. This routine estimates the relative mean displacement signal r_i with respect to the idle milling signal by averaging the signals x_i's,

after discarding the extreme values (e.g., two largest peaks and two smallest peaks), within a moving window. This function is described in equation (2-7). It needs four parameters MLWIN, RWIN, SKIPN, and DIFDL. The parameters RWIN and DIFDL are used to set the initial head pointer offset in the output buffer, which is used as input to the slow wave extractor routine.

Slow wave extractor routine. This routine extracts the slow wave component $S[r_i]$ from the mean displacement signals r_i's by the revolution-oriented residual processing algorithm. This function is implemented based on equations (2-1) and (2-2), using r_i as input. Its output buffer contains the mean displacement signal and the slow wave component, which are read by the detector routine. It needs four parameters NTEETH, RWIN, MAWIN, and GAP.

Residual value extractor routine. This routine extracts the residual signal y_i from the output x_i of the averager routine by the revolution-oriented residual processing algorithm. This function is described in equations (2-1) and (2-2). It needs five parameters NTEETH, RWIN, MAWIN, GAP, and MDWIN. The parameter MDWIN is used to set the initial head pointer in the output buffer, which is used as input to the damage feature extractor routine.

Damage feature extractor routine. This routine extracts the damage feature z_i from the output y_i of the residual value extractor routine, using the median filter. The function eliminates the slowly varying component $S[y_i]$ from the residual signal y_i. The block diagram is shown in Figure 2-5. It needs two parameters MDWIN and NSWIN. The parameter NSWIN is used to set the initial head pointer in the output buffer, which is used as input to the noise suppressor routine.

Noise suppressor routine. In this routine the output sequence $\{z_i\}$ of the damage feature extractor routine is processed to suppress the noisy fluctuations by a relative comparison of amplitudes within a moving window, called NSWIN. This function is implemented based on equations (2-5) and (2-6). When the parameter SRATIO is unity as the default value, this noise suppressor eliminates the current sample unless the absolute value

of the sample is greater than the largest absolute value except the maximum and minimum values within the moving window. Its output buffer contains the noise-reduced feature component w_i, which are read by the detector routine. It needs two parameters NSWIN and SRATIO.

Detector routine. This routine produces the damage feature ratios (i.e., the damage feature to mean displacement signal and the damage feature to slow wave signal) and two states (i.e., the in-process state and the transient state) for both x-axis and y-axis channels. It has two output buffers, called featout and detout. The in-process and transient-phase damage detectors are described in equations (2-8) and (2-9). The in-process and transient states are obtained by thresholding the mean displacement signal r_i and the slow wave signal $S[r_i]$, respectively, with the parameter STH. The processed outputs are sent to the output buffer detout. It also collects the feature data (i.e. the input data to this routine) from both sensory channels in a predefined format of an output buffer featout in order for the Lisp processor to read all the relevant information from the front-end signal processing subsystem. The input data to this routine include the damage feature signal w_i, the mean displacement signal r_i, and the slow wave signal $S[r_i]$ from the x-axis and y-axis channels. It needs three parameters DRDEN, SRDEN, and STH.

Symbolic Processing in a Lisp Environment

The effective representation of domain knowledge is considered to be one of the important factors for the success of a knowledge-based system. The domain of machine tool monitoring exhibits a lot of structure, while still requiring judgemental rules. A viable approach to represent this kind of domain knowledge may be the combination of an object-oriented scheme and a rule-based reasoning: objects can represent the structure of the domain [St86], while production rules can represent the judgemental rules.

The knowledge representation of the object-oriented scheme is implemented as Flavors in the TI Explorer. In this object-oriented programming scheme, state variables are

retained in slots on each instance of a flavor and operations on instances are provided by methods. On the other hand, the production rule facility is implemented as a forward-chaining rule system with a data-driven scheme.

The supervision model applies two basic types of knowledge: One is structured knowledge that will be represented with an object-oriented scheme. It includes an expectation mechanism (e.g., the milling process model, workpiece geometry and tool path) and operation monitor (e.g., state filter and feature interpreter). The other type of knowledge consists of judgemental rules that can be appropriately represented by production rules. Empirical knowledge and judgemental rules (e.g., some feature thresholds and final decision rules) are encoded in this rule-based facility.

Control Structure

This system combines the computational flexibility of event-driven computation in the rule-based reasoning with the strong modularity of object-oriented computation. Objects may be activated synchronously in response to messages received from other objects or from the action part of the rules in the system. This type of activation is the message-passing convention in object-oriented programming environments. Rules are activated in response to certain events that occur during computation. A rule-activating event is represented as a data pattern that is asserted in the system fact base. The rule inference mechanism will execute the rules opportunistically as the situation demands.

The computational activity of the monitoring task in this hybrid knowledge-based system is controlled by a top-level controller that performs repeatedly two different kinds of functions until a quit command is received or a current monitoring task is completed. The first is to handle data and commands that are pending on the local processors and windows. The controller checks whether a quit or pause command is received, and then issue a command to transfer data from the buffers (i.e., featout and detout) on the front-end signal

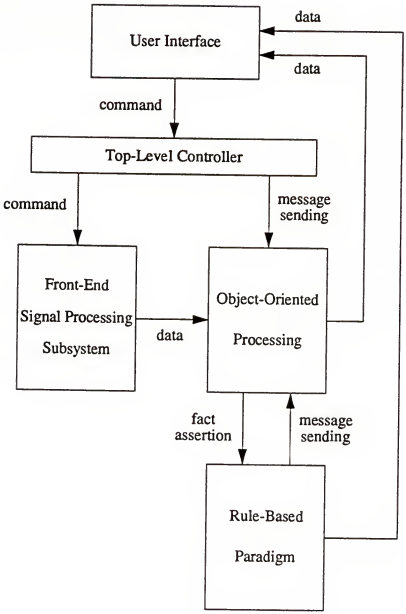

Figure 4-3. The overall control flow of the hybrid knowledge-based system.

processing subsystem to a host buffer on the Explorer memory. The next task is to send a message to the operation monitoring module to execute a sequence of essential tasks that are basically represented in the object-oriented paradigm. These essential activities include tracking the current milling phase (idle, entry, steady state, or exit milling) and observing possible cutter damages during transient and steady state milling phase.

Besides providing explicit control over the sequencing of the essential tasks in the top-level controller, the control structure also effects detailed control within the tasks. It provides a rule inference mechanism, which is normally dormant but may be invoked by an event. Rules may be activated by the methods of some objects during the computation. If the system detects abnormal features or deviation from expected operation during continuous monitoring, it will assert an event into the fact base. The assertion of an event may activate some rules. Figure 4-3 shows the overall control flow in this hybrid knowledge-based system.

Object-Oriented Programming

Object-oriented programming is an approach to software design in which the decomposition of a system is based upon the concept of an object. Objects are entities that combine the properties of procedures and data since they perform computations and save local state. Uniform use of objects contrasts with the use of separate procedures and data in conventional programming [St86].

The Flavor system is an object-oriented programming facility of the Explorer system. In essence, when we define a flavor, we define a data type and a set of operations implemented by the function objects called methods that operate on that data type. The data type defines a set of state variables that record the unique qualities for this data type. The data type instance refers to data objects generated from these flavor definitions, and their associated state variables.

The object-oriented programming method is applied to the implementation of the expectation mechanism and the operation monitoring tasks.

Expectation generation

The expectation mechanism performs based on a priori information prior to the start of machine operation. This module is composed of the milling process model, the state sequence generator, the sensory channel confidence classifier, and the user interface function. Figure 4-4 shows the hierarchical inheritance and instantiation between the objects in the expectation mechanism.

Here we will describe how the objects of the expectation mechanism are represented using Flavors and how they are instantiated. The flavor "cutting-conditions" is defined to represent such variables as spindle-speed and milling-direction. It is inherited by the flavor "operation-expectation." The system requests the cutting parameters from the pop-up menu in the user interface function. The flavor "operation-expectation" is the topmost node of the expectation mechanism hierarchy. It is defined to represent such variables as phase-transition-x, phase-transition-y, and trajectory.

The tool path trajectory is requested from the pop-up menu, and it is instantiated to the slot trajectory of the flavor "operation-expectation." Although the tool path trajectory can be any type, it is currently categorized into three kinds of trajectories: "straight line," "polyline," and "contour." The flavor "straight-line-path" is defined to represent such variables as path-direction, path-length, phase-signals, and workpiece-geometry. The flavor "polyline-path" is defined to represent such variables as path-number, cut-paths, and path-signals. If the trajectory is a polyline path, the number of the straight line path is requested from the pop-up menu. Since a polyline path consists of a multiple of straight lines, the flavor "straight-line-path" is instantiated, as many times as the number of the straight line path, to the slot cut-paths of the flavor "polyline-path." The flavor "contour" is defined to represent any arbitrary trajectory. Presently, if the trajectory is a contour, then we stop the expectation generation.

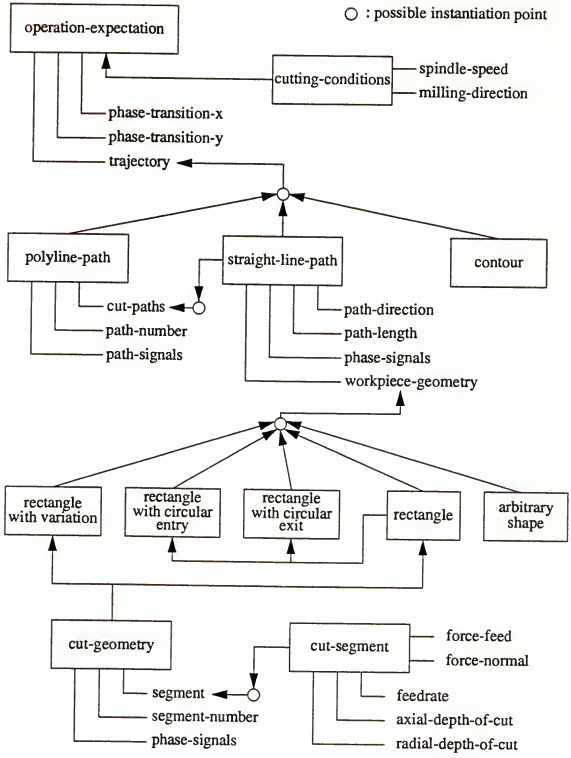

Figure 4-4. Hierarchical inheritance and instantiation of the objects in the expectation mechanism.

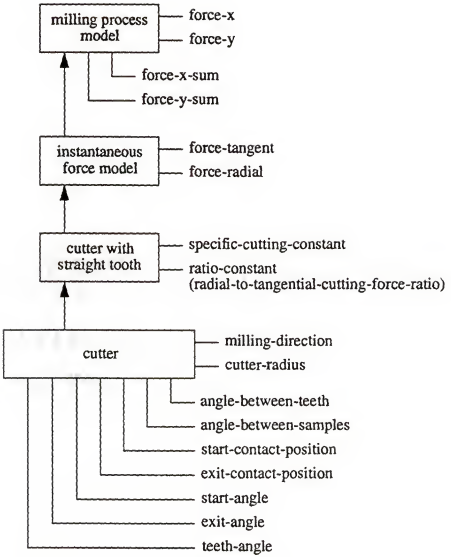

Figure 4-4--continued.

For a straight line path, the variables such as path direction, path length, and workpiece geometry are requested from the pop-up menu, and the workpiece geometry is instantiated to the slot workpiece-geometry of the flavor "straight-line-path." The workpiece geometry means the surface of the swept volume. It is classified into "rectangle," "rectangle with circular entry," "rectangle with circular exit," "rectangle with variation," and "arbitrary shape." The flavor "cut-geometry" is defined to represent such variables as the number of cut segments, cutting parameters in a segment, entry/exit workpiece shape, and phase-signals. The flavor "cut-segment" is defined to represent the cutting parameters in a segment such as feedrate, axial depth of cut, radial depth of cut, force-feed, and force-normal. This flavor is inherited by the flavors "rectangle" and "rectangle-with-variation." The flavors "rectangle-with-circular-entry" and "rectangle-with-circular-exit" are inherited by the flavor "rectangle." The flavor "arbitrary-shape" is defined to represent any arbitrary shape. If the workpiece geometry is an arbitrary shape, then we stop the expectation generation. If the workpiece geometry is a rectangle with variation, the number of cut segment is first requested from the pop-up menu, and the flavor "cut-segment" is instantiated, as many times as the number of cut segments, to the slot segment of the flavor "cut-geometry." For a cut-segment, the feedrate, the axial depth of cut, and the radial depth of cut are requested from the pop-up menu. Then, the flavor "operation-expectation" has obtained all the a priori information for the expectation generation.

We use a milling process model, which is based on an instantaneous force model for the current purpose. This milling process model is described in Chapter 3. In compliance with the configuration of the machine tool, the instantaneous force model is used with a straight-tooth cutter. The straight-tooth cutter inherits information from its component flavor "cutter." The flavor "cutter" is defined to represent several variables related to cutter during the instantaneous force simulation. In order to hold the cutting constants for the tool and workpiece pair, the flavors "cast-iron," "aluminum" and "steel" are defined.

The flavor "operation-expectation" sends the flavor "milling-process-model" a message to calculate the mean displacement signals for each cut-segment. As described in Chapter 3, it can predict a sequence of milling phases and the confidence level attributed to each sensory channel, using the expected mean displacement signals for the x-axis and y-axis channels.

Operation monitoring

Figure 4-5 shows the hierarchical inheritance of the objects in the operation monitoring task. This task consists of a state detector/filter and feature interpreter. The state detector/filter is implemented in the following way. The flavors "milling-state-x" and "milling-state-y" are defined to represent such variables as new-state, old-state, and phase-name for the x and y channels. The two flavors are inherited by the flavors "state-detector-x" and "state-detector-y," respectively. The flavors "state-detector-x" and "state-detector-y" also inherit information from their component flavor "milling-phase." The flavor "milling-phase" is defined to represent such variables as state-to-phase, phase-transition-x, and phase-transition-y. The variable state-to-phase represent the state-to-phase transformation table (for example, a state (1, 1) means the entry phase in milling). The variables phase-transition-x and phase-transition-y represent the state-transition sequences obtained from the expectation mechanism for the both channels. The flavor "state-filter-x" inherits information from its component flavor "state-detector-x." Similarly, the flavor "state-detector-y" also inherits information from its component flavor "state-detector-y." Now we have the state detectors and state filters for both x and y channels.

Feature interpreter. Now we describe the implementation of several flavors for the feature interpreter (Figure 4-5). The extension "-x-exp" means x-channel with expectation, and "-y" means y-channel without expectation. The flavors "damage-feature-x" and "damage-feature-y" are defined to represent such feature variables as current and prior values of the feature-to-mean-displacement-ratio and the feature-to-slow-wave-ratio of the x and y

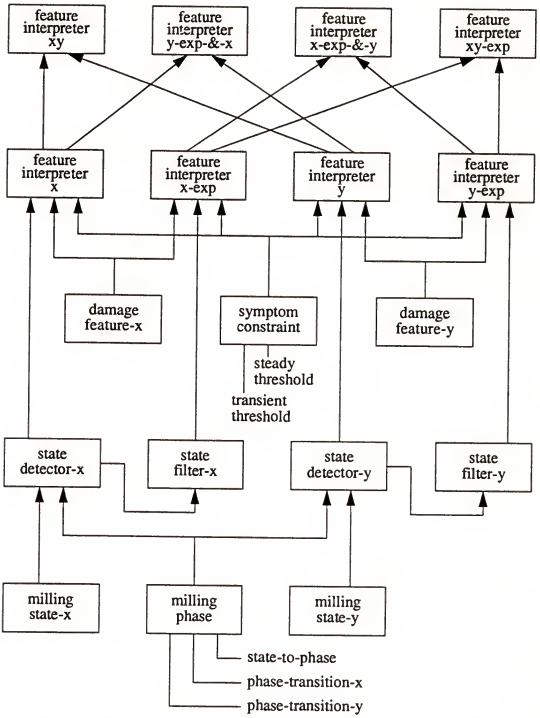

Figure 4-5. Hierarchical inheritance of the objects in the operation monitoring task.

channels. The flavor "damage-feature-x" is inherited by the flavors "feature-interpreter-x" and "feature-interpreter-x-exp." The flavors "feature-interpreter-x" and "feature-interpreter-x-exp" inherit information from their component flavor "state-detector-x" and "state-filter-x," respectively. The flavor "damage-feature-y" is inherited by the flavors "feature-interpreter-y" and "feature-interpreter-y-exp." The flavors "feature-interpreter-y" and "feature-interpreter-y-exp" inherit information from their component flavor "state-detector-y" and "state-filter-y," respectively. All four flavors of the feature interpreter also inherit information from their component flavor "symptom-constraint." The flavor "symptom-constraint" is defined to represent such threshold variables as transient-threshold and steady-threshold. The variable transient-threshold is used to detect the damage feature during the transient (entry and exit) phases. The variable steady-threshold is used to detect the damage feature during the steady-state phase.

For the observation of both x-axis and y-axis channels, the following flavors are defined using the above flavors. The flavor "feature-interpreter-xy" inherits information from its component flavors "feature-interpreter-x" and "feature-interpreter-y." The flavor "feature-interpreter-x-exp-&-y" inherits information from its component flavors "feature-interpreter-x-exp" and "feature- interpreter-y." The flavor "feature-interpreter-y-exp-&-x" inherits information from its component flavors "feature-interpreter-y-exp" and "feature-interpreter-x." The flavor "feature-interpreter-xy-exp" inherits information from its component flavors "feature-interpreter-x-exp" and "feature-interpreter-y-exp." One of the above eight flavors is for the operation monitoring function chosen based on the confidence levels of both sensory channels.

The feature interpreter asserts the damage symptom into the situation assessment module, which is written in a rule-based programming paradigm.

Rule-Based Programming Paradigm

This part discusses one of the most widely used methods of knowledge representation and application, the rule-based programming technique. It also introduces a specific language and techniques for writing programs in this language.

This programming technique is applied in the situation assessment module and some parameter setting rules. The decision rules in the situation assessment module are written around the concept of a three layered panel, which is described in Chapter 3. The complete sets of the parameter setting and decision rules are listed in Appendix B. Since the damage symptom is represented in the form of a quadruplet list (axis, time-index, feature-polarity, milling-phase), patterns of the left-hand side of a decision rule will basically follow the format of the damage symptom. For example, here is a symptom combination rule, which uses primary symptoms' temporal coherency (PSTC):

```
(defrule rule-x-1
    (x ?t1 - ?phase)
    (x ?t2 - ?phase)
    (test (= ?t2 (+ ?t1 *nteeth*)))
=>
    (retract (x ?t1 - ?phase))
    (retract (x ?t2 - ?phase))
    (asserts (x-2 ?t2 - ?phase)))
```

It says that if the feature interpreter of the x-axis channel asserts two primary damage symptoms during the same milling phase with the temporal interval of one revolution, the two symptoms are updated as an intermediate hypothesis.

The rule-based programming paradigm consists of a fact base, a rule base, and an inference engine. These three components will now be described.

Fact base

The fact base stores the current state of knowledge during the problem-solving process by holding symbols that represent facts about the milling domain. The rules act upon these data to update the knowledge about the task being performed.

The items in the fact base are referred to as elements. Syntactically a fact is just a list of symbols. Usually the first component is a predicate with the remaining elements as arguments. A fact is made by executing a statement of the following form:

(ASSERTS (LIST symbol symbol ...) justification-name)

The effect of such a statement is to add to the system's fact base a fact of the form:

(symbol symbol ...)

For example, the following statement asserts that the material of workpiece is cast-iron:

(ASSERTS (LIST 'workpiece 'material 'cast-iron) 'user)

The following fact would be added to the system's fact base:

(workpiece material cast-iron)

with the justification that it was asserted by the user. It is actually the case that ASSERTS simply takes two arguments the first of which should be a list. Any valid Lisp syntax which makes a list is allowable. In particular the following two forms are equivalent:

(ASSERTS '(workpiece material cast-iron) 'user)
(ASSERTS (LIST 'workpiece 'material 'cast-iron) 'user)

The elements in the fact base may be added or removed for a variety of reasons. Creation of fact base elements that represent facts usually follows a major inference. Causes

for removing facts include purposeful forgetting (if a fact is too old to be of interest), garbage collection (if a fact was the result of intermediate computation), and subsumption (another fact contains all the information that the current fact does, so the original fact is now superfluous).

The state of fact base is the main control over the execution of the rule-based program. Control also depends, of course, on the problem-solving strategy embodied in the inference. For instance, which portion of a rule-based program is available for execution at any particular time is determined by the contents of fact base because only those rules that match the fact in fact base are executable; the inference engine determines which rule is actually fired.

Rules

Rules in this paradigm consist of a unique name, a left-hand side that is a sequence of one or more condition elements, and a right-hand side that is a sequence of actions. A typical rule is declared by the following types of statement:

```
(DEFRULE <name>
    <If these conditions exist>
=>
    <Then execute these actions>)
```

Each rule begins with the word DEFRULE followed by the name of the rule. Then comes a list of zero or more conditions. The forward arrow (=>) is the symbol to separate the condition and action portions of a rule.

Left-hand side. The left-hand side of a rule is a Boolean combination of conditions, each of which must be met before the rule can be executed. The allowed Boolean operators are an implied AND function linking all conditions and NOT. Conditions take the form of patterns and tests. Patterns are somewhat arbitrary jumbles of code that are to be matched

against fact base. Tests introduce short sections of procedural code that analyze incoming facts for relationships that beyond the scope of the pattern-matching language.

A pattern has the form:

(symbol term term ...)

where each term is a constant, a wildcard, or a variable. A constant is a Lisp symbol; a wildcard is a symbol of the form, ?; a variable is an expression of the form, ?symbol. A pattern matches a fact if they have identical first components and if terms in the pattern match corresponding symbols in the fact. If the pattern's term is a constant, then the predicate's symbol must be the same constant. If the pattern's term is a wildcard, then the wildcard symbol is interpreted as standing in place of some part of a fact. The wildcard symbol ? matches exactly one component in the fact. If the pattern's term is a variable, then the variable is considered bound to the same symbol. All occurrences of the same variable in any of the patterns of a rule must be bound to the same symbol.

A TEST function allows us to create a temporary Lisp environment as a condition of a rule, in which we can test complex mathematical relationships among data items or perform whatever task the application has forced us to consider. A TEST is a true/false test that is somewhat more complex than a typical predicate.

In addition to the conditions, priority can be assigned to an individual rule. The declaration of priority takes the following form in the left-hand side of a rule:

(DECLARE (PRIORITY number))

where number is a literal number. If a rule has no priority declaration, it is assumed to have a default priority of 0. Priority is our means of giving a rule priority over other rules on the agenda. Within a particular agenda, only one rule will be selected to fire. This will always be the rule, or one the rules, with the highest priority.

Right-hand side. The right-hand side of a rule is composed of a sequence of actions, each of which is a list structure with the name of the action as the first component followed by the arguments to the action. This system has a mechanism for allowing arbitrary functions in Lisp as actions. When all the conditions are satisfied we will call the Lisp procedure, passing it the bindings, the rule-name, and the triggering facts as arguments. There are some predefined action types in this system. The primitive actions that affect the fact base are ASSERTS and RETRACT. The WARN-DAMAGE action is used to output damage information. The argument for the WARN-DAMAGE action is a time tag to be printed.

To assert a new fact, all we do is write out the fact exactly as we want it to appear in the fact base, wrap an ASSERTS phrase around it, and we're all done. It looks like this:

```
(DEFRULE <name>
    <various conditions>        ;required conditions
=>
    (ASSERTS (NEW fact)))       ;assert phrase
```

This ASSERTS phrase would literally assert the fact (NEW fact).

The RETRACT action is a housekeeping and control action. It removes facts from the fact base. Some facts are removed because they slow down the execution of the program appreciably or are no longer needed. You can retract a fact by putting a pattern in a RETRACT phrase.

In addition to the predefined actions, we allow message-sending procedure of the Flavor system as action to send a message to an object. The Lisp function like SETQ can also be used as an action to set the value of a global variable.

Inference mechanism

The inference engine can be described as a finite-state machine with a cycle consisting of three action states: match-rules, select-rules, and execute-rules. This control

mechanism is referred to as the recognize-act cycle. The recognize-act cycle in this rule-based system is similar to the OPS5 [Br85] and ART (Automated Reasoning Tool) [Cl87]. It can be summarized as

```
repeat
    perform match and send rule to agenda
    exit if the agenda is empty
    perform a rule selection
    execute the selected rule
end
```

The rules in a program are treated individually with no ordering relationship imposed on the set of rules, nor is the set divided into subsets for the purpose of selective matching. On each cycle, only the relevant rules are checked for a match of their left-hand side with the fact base. In firing a rule, the ordering of the actions in the right-hand side is important. These actions are executed in the order in which they appear in the selected rule.

The basic concept in this rule-based paradigm is that of data-driven computation. In this system, the facts drive the rules. If all of the conditions of a rule have been satisfied, we say that the system has created an "activation" of the rule. The activation of a rule means that the system has identified a unique collection of facts that satisfy the rule's conditions. The system may find many activations of the same rule at the same time. Activations are sent to the agenda, which is the list of activations currently competing for an opportunity to act.

The second step in each cycle is to select one rule instantiation from the agenda. The next step is to execute the rule. The system evaluates the agenda and executes only the most important activation from the agenda. When the "most important" activation is finally allowed to act, we say that the rule has been "fired." As the rule actions are carried out, changes are made to the fact base, triggering match cycles. Thus the match-rules step is actually

interleaved with the execute-rules step. After firing a rule the system revises the agenda, taking into account any changes in the fact base, and executes the most important activation in the revised agenda. This cycle repeats until the system discovers that the agenda is empty.

A rule enters the agenda only by having its last remaining pattern instantiated by a new fact. A newly asserted fact is the only factor that can activate a rule. There is one exception. A rule can also be activated by the retraction of a fact in certain special circumstance. For instance, a rule designed to fire only in the absence of a certain fact will do so the instant that fact is retracted from the fact base.

This rule inference engine may be invoked by directly asserting an event into the fact base. It can also send a message to the object-oriented paradigm by an action in the form of message-sending procedure. In this way this forward-chaining rule system with event-driven scheme is used in cooperation with the object-oriented paradigm.

Discussions

This chapter has described the implementation of a supervision system for machine tools. The RORPA algorithms have been implemented in an Odyssey board of a TI Explorer, what is a four TMS32020 digital signal processor that interfaces to the NuBus of the computer. The algorithms were written in TMS32020 assembly language code. Both x-axis and y-axis displacement signals can be processed, and we have parallelized the algorithms to take advantage of the multiprocessor architecture of the Odyssey board.

Two methods of the symbolic processing were implemented in the TI Explorer. The object-oriented representations implementing the expectation mechanism and the operation monitoring were written in Flavors, while some feature criteria and final decision rules use a rule-based, forward-chaining scheme.

Next we address the problem of symbolic processing in real-time. It is known to be a very difficult problem [La88]. The problem is associated with the necessity to make a

decision within a predefined time period (in the case of the tool breakage, the decision must be arrived at within a few revolutions). In this system the data rate is basically determined by the sampling rate of raw sensory signals. This rate is directly proportional to the spindle speed due to the synchronous sampling by a 72-tooth gear. Therefore the data rate changes depending on the spindle speed. The system tests showed that the system response time is generally acceptable, but in the high speed milling with spindle speed above 3000 rpm, the computer response time took longer than the actual operation time of the machine tool. One of the reasons may be the speed of the TI Explorer. In addition, the rule shell is not fast enough for this kind of real-time applications.

However, the overall knowledge-based approach can facilitate incremental growth of the system and increase one's productivity. Even though we tried to speed up the symbolic processing by incorporating the object-oriented programming method (for the feature interpreter) with the rule-based paradigm, much research is needed in building special knowledge-based tools for real-time domains.

CHAPTER 5
SYSTEM EVALUATION AND RESULTS

The developed system was tested with 194 sets of test data generated from a White Sundstrand Series 20 Omnimil, a milling machine used for research in the Machine Tool Laboratory of the Mechanical Engineering Department at the University of Florida. The test data was collected by one of Dr. Jiri Tlusty's graduate students, Mr. Tarng. The experimental data were obtained at several different speeds, radial depths of cut, axial depths of cut, workpiece geometry variations, cutting path changes, cutters with different number of teeth, and different workpiece materials.

In this chapter we will present experimental results that illustrate the validation of the proposed numeric/symbolic supervision model. We will first describe the data collection set-up, and next the results of the system performance. From the results of the tests, a general perspective to problems can be illustrated with discussions on the system performance, and ways to further improve the system.

Data Collection Set-up

A Sundstrand Omnimil 20 machining center (mfg. White Sundstrand Machine Tool Co., IL), with a maximum rated spindle speed of 5500 rpm and a rated horse power of 25 at 1200 rpm was used in the experiments. It is a horizontal spindle machine with three linear axes, a 360 discrete degree rotary index table, and a 30 position automatic tool exchanger [Ta88].

All of the face milling test data were obtained by using a 4-inch diameter double-negative ZN3M Carbaloy milling cutter (GE Co.) with the catalog number SNG634 or

SNG632 of eight silicon nitride inserts. The run-out on the face milling cutter was adjusted within 0.0254mm (0.001"). For the end milling, a 19.05mm (0.75") diameter carbide end mill cutter (Robb Jack Co.) with 3 and 4 flutes of 30 degree helix angle was used. In addition to this, three kinds of workpiece material, cast-iron GM-241, aluminum 7075-T7351, and steel 4330 were used for cutting tests.

The displacement signal was collected by a multichannel non-contact measurement device, Accumeasure system 1000 (MTI Co.). This system determines the capacitance between a capacitance probe with a sensing electrode and a ground target via a coaxial connecting cable. Then the capacitance is transmitted to a probe amplifier module (AS-1023PA) which has a corner frequency of 5 KHz. An analog signal proportional to the probe-to-target gap is generated in the circuit when a resultant variable capacitance appears. In addition, an anti-aliasing filter was used, with a cutoff frequency of 1 KHz, in order to reduce the effect of aliasing that may occur in the sampling process.

One capacitance probe is located in the x-axis direction, and the other is located in the y-axis direction. The probes are located between the housing and spindle close to the its flanges (see Figure 5-1). For the front bearings of the spindle, an automatic mist oil spray system is provided in the machining center. However, the lubrication oil flows through the gap between the probe and the spindle when the spindle rotates. This may cause a little measurement error in the displacement signal. The displacement signals were recorded on an IBM PC-AT microcomputer using a data acquisition system (Data Translation DT2812). The data acquisition system provides four A/D converters with a maximum simultaneous sampling rate of 27.5 KHz and two D/A converters with a maximum rate of 33.0 KHz.

In order to sample the displacement signal synchronized with the spindle rotation, external sampling and triggering were used. A magnetic pick-up sensed the distance between itself and a steel gear with 72 teeth mounted on the spindle. When the spindle rotated, the distance was converted into an analog signal. This analog signal was

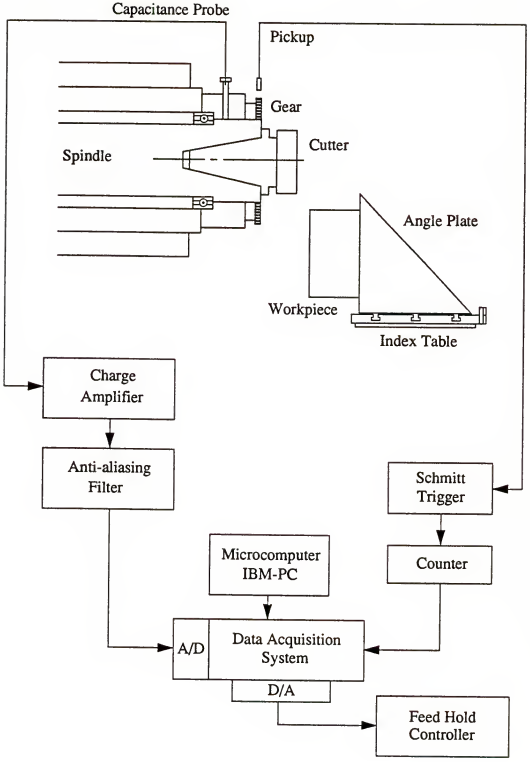

Figure 5-1. Schematic diagram of the data collection set-up.

transmitted through a comparator, a Schmitt trigger, and counters from which the external sampling and triggering signals were obtained. Figure 5-1 shows the schematic diagram of the experimental set-up for the displacement measurements [Ta88].

Evaluation Results

The system performance is ultimately measured in terms of detection and false positive rate for all the test cases. The test data were obtained under various operating conditions for three kinds of workpiece materials (i.e., cast iron, aluminum and steel) on a White Sundstrand Series 20 Omnimil, a milling machine. The test data were read into the Lisp workstation (Explorer) and stored on files. The test data contained different cutting conditions: spindle speeds from 400 to 3750 rpm, axial depth of cut from 0.04 to 0.1 inch, up and down milling, radial immersions from 1/8 to 1/1, rectangular and slant workpieces, machining over a slot, good and broken cutters, for a total of 194 cases.

Below, the test data obtained from different cutting conditions are described. All the data for induced tool damage cases were obtained under controlled conditions with total or partial damage on the inserts, except in one case when the tool broke during milling of a steel workpiece. The test data were classified into three groups based on the workpiece material.

Data for the first group were obtained using the workpiece material of cast-iron. This group consists of 154 cases, being 89 undamaged cases and 65 damaged cases. Table 5-1 summarizes the test results for the up milling cases with the cutting parameters of rectangular workpiece boundary, 8 teeth and feed per tooth of 0.01". In the table, the number in the parenthesis under the undamaged cases indicates the number of false positive cases. Both false alarm cases showed just one occurrence of false alarm.

Table 5-1. The up milling cases with feed per tooth of 0.01".

Spindle Speed (rpm)	Axial Depth of Cut (inch)	Radial Immersion	Number of Cases	
			Undamaged	Damaged
600	0.05	1/8	1	1
		1/4	2	1
		1/2	4	1
		3/4	1	1
		1/1	1	1
	0.06	3/4		1
	0.1	1/4	1	1
		1/2	1	1
		1/1	1	1
1425	0.05	1/2	1	1
1455	0.05	1/2	1	
		1/1	1	
1485	0.05	1/2	1	1
1500	0.1	1/4	1	1
		1/2	1	1
1515	0.05	1/2	1	
1845	0.1	1/2	1	1
1860	0.1	1/4	1	1
1935	0.1	1/2	1	1
1965	0.05	1/2	2	1
	0.1	1/4	1	1
2160	0.05	1/2	1	1
2175	0.05	1/2	1	
	0.1	1/4	1	1

Table 5-1--continued.

Spindle Speed (rpm)	Axial Depth of Cut (inch)	Radial Immersion	Number of Cases	
			Undamaged	Damaged
2250	0.1	1/2	1	1
3000	0.05	1/2	2	1
	0.1	1/4	1	1
3555	0.1	1/4	1	1
3570	0.05	1/2	1	1
3585	0.05	1/2	3	
	0.1	1/4	1	
3630	0.05	1/4	2 (1)	1
	0.1	1/4	3 (1)	1
3690	0.1	1/4	1	1
3750	0.05	1/2	1	2

Table 5-2. The down milling cases with feed per tooth of 0.01".

Spindle Speed (rpm)	Axial Depth of Cut (inch)	Radial Immersion	Number of Cases	
			Undamaged	Damaged
600	0.05	1/2	2	3
	0.06	3/4		1
	0.1	1/4	1	
		1/2	2	1
1830	0.05	1/2		1
1860	0.05	1/2	1	
1920	0.05	1/2		1
1950	0.05	1/2	1	
3000	0.05	1/2	1	1
3375	0.05	1/2	4	2

Table 5-3. The cases of sudden changes of axial depth of cut.

Milling Direction	Axial Depth of Cut (inch)	Number of Cases	
		Undamaged	Damaged
up	0.05 + 0.1 + 0.04	1	
	0.04 + 0.1 + 0.05	1	1
down	0.04 + 0.1 + 0.05	1 (1)	
	0.05 + 0.1 + 0.04		1

The down milling cases with the cutting parameters of rectangular workpiece boundary, 8 teeth and feed per tooth of 0.01" are given in Table 5-2. The down milling cases show perfect recognition of damaged cutters as well as no false alarm error.

Table 5-3 presents the cases of sudden changes of axial depths of cut with the cutting parameters of rectangular workpiece boundary, 8 teeth, spindle speed of 600 rpm, radial immersion of 1/2 and feed per tooth of 0.01". There was one false positive decision in the above cases. Following are similar cases with the cutting parameters of up milling, workpiece boundary of rectangle, 8 teeth, radial immersion of 1/4 and feed per tooth of 0.01" (Table 5-4):

Table 5-4. The cases of sudden changes of axial depth of cut at high spindle speed.

Spindle Speed (rpm)	Axial Depth of Cut (inch)	Number of Cases	
		Undamaged	Damaged
3375	0.05 + 0.1 + 0.04	2	

Feedrate is one of the important cutting parameters related to tool life. The different feedrate cases with the cutting parameters of workpiece boundary of rectangle, 8 teeth, spindle speed of 600 rpm, axial depth of cut of 0.05" and radial immersion of 1/2 are tabulated in Table 5-5. The data for the up milling cases were given with the varying feedrates. There was perfect recognition of damaged cutters and no false alarms in these data sets.

The changes of depth of cut can be divided into two kinds: one is sudden change of depth of cut, which is shown in the previous cases (see Tables 5-3, 5-4, and 5-5), and the other is gradual change of depth of cut. Table 5-6 describes the test results for the cases of gradual changes of radial immersion with the cutting parameters of up milling, varying

Table 5-5. The different feedrate cases.

Milling Direction	Feedrate (inch per min)	Number of Cases	
		Undamaged	Damaged
up	36 + 72	1	1
	72 + 36	2	2
	36 + 72 + 48 + 60	3	3
down	24		1
	72		1

Table 5-6. The cases of gradual changes of radial immersion.

Start and Exit Angle Radial Depth of Cut (Deg)	Number of Cases	
	Undamaged	Damaged
0 - 180	1	1
180 - 0	1 (1)	1

Table 5-7. The cases of gradual changes of axial depth of cut.

Milling Direction	Axial Depth of Cut (inch)	Number of Cases	
		Undamaged	Damaged
up	0.1 - 0.05	2	1
	0.05 - 0.1	1	1
down	0.1 - 0.05	1	1
	0.05 - 0.1	1	1

immersions of cut, 8 teeth, spindle speed of 600 rpm and axial depth of cut of 0.05". A false alarm case happened with just one occurrence of false alarm during the steady milling phase. Next, the cases of gradual changes of axial depths of cut with the cutting parameters of 8 teeth, spindle speed of 600 rpm and radial immersion of 1/2 are given in Table 5-7. In these cases we do not use the expectation mechanism since it is very difficult to describe the workpiece geometry.

Table 5-8 shows the test results for the cases of interrupted cutting over a slot with the cutting parameters of 8 teeth, axial depth of cut of 0.05" and spindle speed of 600 rpm.

Table 5-8. The cases of interrupted cutting over a slot.

Milling Direction	Radial Immersion	Number of Cases	
		Undamaged	Damaged
up	3/4	2	1
face	1/2	2	1

Table 5-9. The cases of interrupted cutting over circular entry or exit.

Interrupt	Number of Cases	
	Undamaged	Damaged
circular entry	1	
circular exit	1	

In addition, the cases of interrupted cutting over circular entry and exit with the cutting parameters of up milling, 8 teeth, spindle speed of 600 rpm, axial depth of cut of 0.1", radial immersion of 1/2 and feed per tooth of 0.01" are given in Table 5-9.

Chipping occurs when the cutting edge is broken off only partly rather than being totally broken. This makes the cutting edge ragged and inefficient. As a result, the total breakage of cutting edge usually follows very quickly. Table 5-10 presents the test results for the cases of partial and total breakage of inserts with the cutting parameters of rectangular workpiece boundary, up milling, axial depth of cut of 0.05", radial immersion of 1/2 and feed per tooth of 0.01". The supervision system recognized all the damages, including partially broken cutters.

Table 5-10. The cases of partial and total breakage of inserts.

Percentage of Breakage of Insert	Number of Cases	
	Undamaged	Damaged
100%		3
50%		3

Table 5-11. The 4-flute end milling cases.

Axial Depth of Cut (inch)	Feed per Tooth (inch)	Number of Cases	
		Undamaged	Damaged
0.05	0.005	1	
	0.01	1	
0.1	0.005	1	

The 4-flute end milling cases with the cutting parameters of spindle speed of 400 rpm, radial immersion of 1/1, and 3/4" diameter carbide cutter are given in Table 5-11. And Table 5-12 shows the 3-flute end milling cases with the cutting parameters of spindle speed of 600 rpm, radial immersion of 1/1, feed per tooth of 0.01" and 3/4" diameter carbide cutter. In the 3-flute end milling cases, there was one false alarm.

Table 5-12. The 3-flute end milling cases.

Axial Depth of Cut (inch)	Number of Cases	
	Undamaged	Damaged
0.05	3 (1)	
0.1	1	

A corner slotting case with the cutting parameters of end milling, 3 flutes, spindle speed of 600 rpm, axial depth of cut of 0.1" and feed per tooth of 0.01" was tested using undamaged 3/4" diameter carbide cutter. There was no false alarm in this case. In addition, a slotting case with the cutting parameters of 4 flutes, spindle speed of 600 rpm, radial immersion of 1/1, axial depth of cut of 0.05" and feed per tooth of 0.01" was tested using a damaged 1" diameter HSS cutter (grinding wear). The system recognized the damaged tool immediately.

The test results of this group indicate 5 false alarm cases out of 89 undamaged cases and perfect recognition of 65 damaged cases with total and partial breakages.

A second group of test data was obtained from an aluminum workpiece. This group comprises a total of 17 cases, which consist of 10 undamaged and 7 damaged cases. A face milling cutter was used with SNE-633 carbide inserts for this group of test cases. Table 5-13 presents the test results for the aluminum workpiece cases with the cutting parameters

Table 5-13. The aluminum workpiece cases.

Milling Direction	Axial Depth of Cut (inch)	Radial Immersion	Number of Cases	
			Undamaged	Damaged
up	0.05	1/2	4	1
	0.1	1/4	1	1
		1/2	1	1
		3/4	1	1
		1/1	1	1
down	0.05	1/2	1	1
	0.1	1/2	1	1

Table 5-14. The steel workpiece cases.

Milling Direction	Axial Depth of Cut (inch)	Radial Immersion	Number of Cases	
			Undamaged	Damaged
up	0.05	1/8	1	1
		1/4	1	1
		1/2	1	1
		3/4	1	1
		1/1	1	1
	0.1	1/2	1	1
down	0.05	1/2	1	1
	0.1	1/2	1	

of rectangular workpiece boundary, 8 teeth, spindle speed of 600 rpm and feed per tooth of 0.01". The tests using this group of data show no false positive errors for 10 undamaged cases and perfect recognition of 7 damaged cases.

A third group of test data was obtained from a steel workpiece. In this group, a total of 23 cases, consisting of 15 undamaged and 8 damaged cases, were obtained. A face milling cutter was used with SNE-633 carbide inserts for this group of test cases. Table 5-14 describes the test results for the steel workpiece cases with the cutting parameters of rectangular workpiece boundary, 8 teeth, spindle speed of 600 rpm and feed per tooth of 0.01".

In addition, further sets of data that were obtained captured the natural tool breakages. These sets with the cutting parameters of up milling, rectangular workpiece boundary, 8 teeth, spindle speed of 600 rpm and radial immersion of 1/2 are given in Table 5-15. Although the damaged cases had a chain of three tool breakages during milling of the steel workpiece, all three events of tool breakage were recognized immediately.

Table 5-15. The steel workpiece cases that captured the natural tool breakage.

Axial Depth of Cut (inch)	Feed per Tooth (inch)	Number of Cases	
		Undamaged	Damaged
0.05	0.015	2	
	0.02	2	
0.1	0.01	3	
	0.015		1

Table 5-16 presents the overall results of the tests. In each row two numbers are shown. The first is the number of cases and the second is the number of detected breakage cases. The system has no false negatives, i.e. all the damaged cases were accurately detected. Only 5 false positives were detected, and they all occurred with the workpiece material

of cast-iron. This seems natural since the cast-iron cases were more used than the others. Two of the false alarms occurred in high speed of milling (3630 rpm). Another occurred in down milling when a sudden change in depth of cut was encountered. Another false alarm occurred on a slanted workpiece. And finally another false alarm was detected in a 3-flute end-milling operation.

Table 5-16. The overall results of the tests.

Workpiece	Undamaged	Damaged	Total
Cast-iron	89 / 5	65 / 65	154
Aluminum	10 / 0	7 / 7	17
Steel	15 / 0	8 / 8	23
Total	114 / 5	80 / 80	194

Discussions

The automated supervision system was tested and evaluated with a large amount of data from a wide range of operating conditions by incrementally incorporating relevant rules into the system's knowledge base. The test data contained different cutting conditions: spindle speeds from 400 to 3750 rpm, axial depth of cut from 0.04 to 0.1 inch, radial immersions from 1/8 to 1/1, feed per tooth from 0.005 to 0.02 inch, up and down milling, rectangular and slant workpieces, sudden changes of axial depth of cut, sudden changes of feedrates, machining over a slot, interrupted cutting with circular entry or exit, corner slotting, face and end milling, and cast-iron, aluminum and steel workpieces.

The overall results showed that all the damaged cases, which included totally broken, chipped or worn cutters, were recognized without a missed-detection. On the other hand, there were only 5 false alarm cases out of 114 sets of test data from undamaged

cutters. These results corroborate our opinion that a numeric processing front-end integrated in a symbolic environment to refine detection criteria is a viable approach to tool breakage detection. The signal processing parameters were basically established by evaluating the system with typical sets of operation data (5 broken and 5 good cutter cases) shown in Chapter 2.

It was observed during the system test that a severe noise fluctuation is embedded in the displacement signal collected from the high speed milling with the spindle speed greater than 3100 rpm. This frequency range is near the resonant frequency of the machine tool (see Figure 2-6). Thus, we used the higher detector threshold value in the spindle speed falling in this range.

Although the number of test cases included in the present analysis is sufficient to provide a basic system performance validation, there are many more possibilities involving mainly workpiece geometries and trajectory changes that must be analyzed to improve and augment the capabilities of the supervision system.

A total of 51 rules are included in the final decision knowledge base, and these rules are listed in Appendix B. The developed system has the flexibility to allow modification and incorporation of more rules in the system. Potentially, there exists room for further performance improvement by elaborating on the system's knowledge base with a compromise between minimum decision time and false alarm rate.

CHAPTER 6
CONCLUSION

This chapter will present a summary of the main ideas described in this dissertation and suggest directions for further research in machine tool supervision.

Summary of Main Ideas

In this dissertation we have developed a sensor-based computer-assisted supervision system, which detects and identifies totally broken, chipped or worn cutters. The tasks in this research can be divided into three categories: development of algorithms and a supervision model, implementation of the algorithm and model in a multiprocessor environment, and system evaluation using a variety of real data.

A new real-time algorithm (revolution-oriented residual processing algorithm--RORPA) has been developed for automatic detection of cutter damages in machine tools. The method processes sensory signals and extracts features from x-axis and y-axis displacement sensors attached to the headstock of the milling machines. The RORPA can separate the damage features from the run-out signal and other reference signals produced by normal machining. The residual sequence represents changes in the system states. The algorithm separates the component corresponding to normal operation from the component that may contain the features of damaged cutter. The mean displacement signals are used as reference information for the detection criterion. The RORPA enhances the features of tool damage, even in the presence of noise, transient and run-out effects. Results of damage detection, based on the experimental data, show the RORPA to be very effective and sensitive in identifying operational states of a milling machine.

In addition, a knowledge-based supervision model has been developed and applied to enable robust, accurate automated decisions. The model hierarchically integrates the RORPA in a knowledge-based processing environment where rules and objects co-exist. The numeric/symbolic model, which incorporates physical models and empirical knowledge, consists of four tasks: expectation generation, operation monitoring, situation assessment and action execution. The model was developed not only to address the tool breakage problem, but also to include other, multi-sensor, information as will be required for machine tool supervision.

The two aspects of supervision model have been implemented in the Texas Instruments Explorer I, a Lisp workstation. In the symbolic environment the object-oriented representation implementing the expectation mechanism and the operation monitoring are written in Flavors, an object-oriented programming facility. The situation assessment uses a rule-based, forward-chaining scheme. The RORPA has been implemented in a multiprocessor board, which consists of four TMS32020 digital signal processors interfaced to the NuBus of the computer.

We have empirically validated the proposed supervision model. The system evaluation was performed using a variety of real data cases (194 cases). The overall results indicate that all 80 damaged cases, which included totally broken, chipped or worn cutters, were recognized without a missed-detection. On the other hand, there were only 5 false alarm cases out of the 114 sets of test data from undamaged cutters.

The outcome of this research can impact on the productivity of manufacturing industry through the automated tool damage detection of machine tools. The system evaluation results show that our approach can be a viable alternative to the human supervision of machine tools. The main contributions of this dissertation are: development of a new algorithm (RORPA) using the displacement signal for the tool damage detection in milling; development of a numeric/symbolic model to incorporate physical models and

empirical/experiential knowledge; implementation of the automated supervision system in a multiprocessor architecture.

Further Work

In this dissertation we have developed a supervision system for machine tools. The system evaluation has been performed using a variety of real data. Machine tool supervision with the proposed numeric/symbolic model gave very good results, but further studies are still needed. Although the number of test cases included in the present evaluation is sufficient to provide a basic system performance validation, there are many more possibilities involving mainly workpiece geometries and trajectory changes that must be analyzed to improve and augment the capabilities of the supervision system.

This work is addressing the implementation of the supervision model as a real-time system. The front-end subsystem in its current design is able to react in real-time. The bottleneck is the symbolic processing. The goal is to detect the broken cutter within a few revolutions. However, it is very hard to guarantee that the symbolic processing will arrive at a decision within this time limit, due to the incremental way that symbolic reasoning works. In fact, symbolic reasoning is a fact gathering approach. In certain instances the sensory signal may be so clear-cut that an almost immediate decision is made. In other cases, lots of inferences may need to be performed which increases the decision time. One direct way of improving the situation is to speed up the symbolic environment (i.e., more inferences per unit time), but other strategies are also possible, such as deciding with incomplete knowledge. Another promising approach would be to use neural networks to interpret the sensory channels.

In this dissertation we have focused on the tool damage detection problem using the displacement sensors as sensory channels. It would be helpful to utilize sound captured by a directional microphone over the machine tool to monitor machining states. It could be a

good approach to integrate the sound signal with the displacement signal for the tooth breakage detection, and to extend the sound processing to other aspect of machine supervision such as chatter detection.

APPENDIX A
DESCRIPTION OF TOOLKIT FOR THE EXPLORER AND TMS32020 ODYSSEY BOARD

This toolkit is fully described in [Ma88]. Here we describe the functions used in this research in order to help understand the implementation of the signal processing algorithms in a multiprocessor system.

The onboard execution of user selected functions is controlled by a kernel which is included in each load module which is downloaded to the board. The kernel processes all interrupts and accepts the commands (execute and quit) from the Explorer. As part of the execute command, the kernel receives a list of addresses of the functions to be executed along with associated communication block addresses. The kernel sets up the communication block address and calls the first function on the list, when control is returned, the next communication block is set up and the next function is called. This is continued until the list is processed. If the list was executed with the repeat flag set, then the list is repeated until a quit command is received. The repeat command causes the complete list to be repeated. When the kernel receives the quit command, the function being processed is terminated and control is returned to the kernel command loop.

The kernel program for the master processor is different from the kernel program for the slave processors. The master kernel needs to be able to process the CODEC interrupts and perform the basic function of the master controller program.

The kernel is normally the only module to communicate (receive commands and transmit completions) with the host computer. The kernel is designed to support time sharing (giving each function to be executed a time slice) of the tool kit functions. Time sharing is a user selectable option. This time sharing is not used in the present application.

No attempt is made by the tool kit support routines to optimize Odyssey resources. The user is required to divide the task into load modules and specify in which processor each module will execute. The support routines will allocate data memory in the processors in accordance with the user directives.

Support Routines

The Lisp support routines used in this research can be divided into two categories: runtime and non-runtime.

Runtime Support Routines

The runtime support routines will include routines to allocate buffers, define tasks, download modules, fill buffers, empty buffers, start task execution, and stop task execution.

DEF-LIBRARIES routine. This routine must be executed before any other of the toolkit runtime routines are used. It requires the user to specify the object libraries to be searched when defining a load module with the DEF-MODULE routine. The DEF-LIBRARIES routines will read the specified libraries into memory along with their corresponding control files and initialize internal toolkit support tables. Also, the kernel routines from directory 'odyssey-host:tools;' will be read into the Explorer memory buffers. Up to 16 libraries may be accessed by this routine.

DEF-MODULE routine. This routine will allow the user to build a load module which may be downloaded and executed on the Odyssey processor. A load module is one or more functions which will be combined with the kernel routine into a downloadable format. The kernel is automatically include as the first routine in the module. The DEF-MODULE does not logically link resources. It physically combines functions so that they may reside in the same processor at the same time. Functions may be logically combined by

sharing buffers and having the order of their execution specified in the runtime execution routine.

LOAD-MODULE routine. This routine controls the downloading of the load module to the processor program memory. Any information in the processor's program memory may be overwritten by the load module.

DEF-BUFFER routine. This routine allows the user to define data buffer space on the Odyssey board. Buffer space is allocated by processor id and a unique buffer name. The input includes the processor id, buffer name, and size of buffer. All buffer locations will be from external memory which can be accessed by other processors. This routine will add a fixed amount of overhead space (for buffer control) and allocate the space in the specified data memory (if it will fit) and initialize the buffer control information to specify that the buffer is empty. Before a buffer may be referenced in any other toolkit routine it must be defined using this routine.

PUT-DATA routine. This routine transfers data from the host buffer on the Explorer memory to a buffer on the Odyssey board. It requires the name of the Explorer buffer, the amount of data, and the name of the Odyssey input buffer. This routine will check the control information on the buffer and transfer only up to the amount of data that will fit into the buffer. If the wait flag is set, control is not returned to the calling program until all the data has been transferred. Otherwise, control is returned when the buffer is filled or there is no more data available. After the transfer, this routine updates the head pointer in the Odyssey buffer.

GET-DATA routine. This routine transfers data from the buffer on the odyssey board to a host buffer on the Explorer memory. This requires the name of the Odyssey output buffer, the amount of data, and the name of the Explorer buffer. This routine will check the control information on the buffer and get the requested amount or empty the buffer, whichever is less. If the wait flag is set, control is not returned to the calling program until the requested amount is transferred from the Odyssey buffer. Otherwise, control is returned

when the buffer is empty or there is no more data available. After the transfer, this routine updates the tail pointer in the Odyssey buffer.

DEF-PARAMETERS routine. This routine specifies which input/output buffers a routine will use and the routine parameters (if any) which will be used when the routine is executed. The user must call this routine to define and transmit (to the Odyssey board) the parameters for each routine before it can be executed. If the user does not include the routine parameters in the call, the set of default parameters from the control file will be used. This routine allocates the memory space for the communication block of the routine.

EXECUTE-MODULE routine. This routine initiates the execution of a function or a sequence of functions. After execution has been started, control is returned to the calling program. Execution of the same sequence of functions in the same load module in different processors may be accomplished by a single call to EXECUTE-MODULE. The user may combine the execution of several functions within the load module. The user may also have a sequence of functions repeatedly executed. The load module must currently exist in the processor and the functions parameters set before the execute command is executed. Up to 16 functions can be executed with a single call to EXECUTE-MODULE.

QUIT-TASK routine. This routine terminates the execution of the functions which was initiated by the EXECUTE-MODULE routine. Control in the specified processors is returned to control kernel command loop.

QUIT-PROGRAM routine. This routine should be called before termination of application program to release the Explorer and Odyssey resources which were allocated by the tool kit routines. If the argument to QUIT-PROGRAM is true, all resources are released. If the argument to QUIT-PROGRAM is nil, the system is returned to the state it was in after the call to DEF-LIBRARIES was completed.

Non-Runtime Support Routine

There are two non-runtime routines which are used to maintain and update the toolkit libraries and associate control files.

LIBRARY-320 routine. This routine allows the application developer to add, replace, delete, extract, and list routines in an existing library or create a new library. It is executed by typing in "(library-320)" under the Lisp Listener.

UPDATE-CONTROL routine. This routine allows the application developer to add, delete, or replace control information in an existing control file or create a new control file. A control information file should have one entry for each function in the associated object library file. The control file will be a text Lisp file which will be compiled by UPDATE--CONTROL and the compiled file will be loaded and executed by the DEF-LIBRARIES routine. The following information is included in each entry: function name, maximum size in words of working space required, internal memory flag (set if the function has been written to use specific pages of memory), number of input buffers (up to four allowed), minimum size of each buffer, number of output buffers (up to four allowed), minimum size of each buffer, number of parameter words, and default values for parameters (up to 32 allowed). The UPDATE-CONTROL routine is executed by typing in "(update-control)" under the Lisp Listener.

Function Alias

In order to support the execution of the same function (with different parameters/buffers) in a sequence, the use of aliasing is permitted. An alias is the function name followed by "#n" where n is a single digit number. Up to 10 alias may exist for a function. The use of alias functions is permitted as arguments for the DEF-PARAMETERS and EXECUTE-MODULE routines. If an alias is used as one of the arguments to the execute routines, the parameters must have been defined for that alias by using the

DEF-PARAMETERS routine. For example, execution of the sequence ("A" "A#1") executes the same "A" function both times but each execution uses a different communication block to get the parameters and buffer addresses.

Executing Library Functions

The library functions may be downloaded and executed on the Odyssey board in several different input/output modes and the functions may be stand-alone functions or part of system of functions. The input may be real-time (CODEC), file mode (from Explorer), user generated (from Explorer), or generated by another Odyssey function. The output may be real-time (CODEC), file mode (to Explorer), user requested (to Explorer), or provided to another Odyssey function as input. Therefore, in order to operate in all of the above modes a very general input/output interface is required.

In general each function downloaded will have a memory address of a communication block. The communication block will contain all the information which the function requires to run. This information includes addresses of input/output buffers and any control parameters which the function needs to execute.

A control kernel will be included in each load module in each processor which the user is using for a library function. Initial control is passed to the kernel. The kernel waits for inputs from the Explorer. The kernel's execute command consists of a list of functions (addresses) to execute and the associated communication blocks. It in turn, calls the functions on the list. When a function terminates it returns to the control kernel which will call the next function.

The application program is responsible for providing or setting up the input data for the system. If real time data is required, the application program must set up, start, and stop the data.

APPENDIX B
DECISION AND PARAMETER SETTING RULES

```
;;; -*- Mode:Common-Lisp; Package:USER; Base:10 -*-
;;
;;
;; This part of rule base is to decide damage
;;

;;;
;;; First layer for two symptoms integration: Symptom Panel
;;;

;; negative-negative (pi) pair in x-axis: PSTC

(defrule rule-x-1
    (x ?t1 - ?phase)
    (x ?t2 - ?phase)
    (test (= ?t2 (+ ?t1 *nteeth*)))
 =>
    (retract (x ?t1 - ?phase))
    (retract (x ?t2 - ?phase))
    (asserts (x-2 ?t2 - ?phase)))

;; negative-positive (p) pair in x-axis: ASDP

(defrule rule-x-2
    (x ?t1 - ?phase)
    (x ?t2 + ?phase)
    (test (= ?t2 (+ ?t1 1)))
 =>
    (retract (x ?t1 - ?phase))
    (retract (x ?t2 + ?phase))
    (asserts (x-2 ?t1 perfect ?phase)))

;; negative-negative (2t) pair in x and y axis: PSIS

(defrule rule-xy-1
    (x ?t1 - ?phase)
```

```
    (y ?t1 - ?phase)
 =>
    (retract (x ?t1 - ?phase))
    (retract (y ?t1 - ?phase))
    (asserts (xy ?t1 ?phase)))

;; negative-negative pair in y-axis: PSTC

(defrule rule-y-1
    (declare (priority 10))
    (y ?t1 - ?phase)
    (y ?t2 - ?phase)
    (test (= ?t2 (+ ?t1 *nteeth*)))
 =>
    (retract (y ?t1 - ?phase))
    (retract (y ?t2 - ?phase))
    (asserts (y-2 ?t2 - ?phase)))

;; negative-positive pair in y-axis: ASDP

(defrule rule-y-2
    (y ?t1 - ?phase)
    (y ?t2 + ?phase)
    (test (= ?t2 (+ ?t1 1)))
 =>
    (retract (y ?t1 - ?phase))
    (retract (y ?t2 + ?phase))
    (asserts (y-2 ?t1 perfect ?phase)))

;;;
;;; Second layer for symptom and first layer integration: Intermediate Panel
;;;

;; negative symptom in x-axis: PSTC

(defrule rule-x-21
    (x-2 ?t1 ?shape ?phase)
    (x ?t2 - ?phase)
    (test (= ?t2 (+ ?t1 *nteeth*)))
 =>
    (retract (x-2 ?t1 ?shape ?phase))
    (retract (x ?t2 - ?phase))
    (asserts (x-3 ?t2 ?shape ?phase)))

;; positive symptom in x-axis: ASDP
```

```
(defrule rule-x-22
    (x-2 ?t1 - ?phase)
    (x ?t2 + ?phase)
    (test (= ?t2 (+ ?t1 1)))
 =>
    (retract (x-2 ?t1 - ?phase))
    (retract (x ?t2 + ?phase))
    (asserts (x-3 ?t1 + ?phase)))

;; pair in x-axis and negative symptom in y-axis

(defrule rule-xy-21
    (x-2 ?t1 - ?phase)
    (y ?t1 - ?phase)
 =>
    (retract (x-2 ?t1 - ?phase))
    (retract (y ?t1 - ?phase))
    (asserts (xy-3 ?t1 - - ?phase)))

;; 2t pair and negative symptom in x-axis

(defrule rule-xy-22
    (xy ?t1 ?phase)
    (x ?t2 - ?phase)
    (test (= ?t2 (+ ?t1 *nteeth*)))
 =>
    (retract (xy ?t1 ?phase))
    (retract (x ?t2 - ?phase))
    (asserts (x-2-y ?t2 - - ?phase)))

;; 2t pair and positive symptom in x-axis

(defrule rule-xy-23
    (xy ?t1 ?phase)
    (x ?t2 + ?phase)
    (test (= ?t2 (+ ?t1 1)))
 =>
    (retract (xy ?t1 ?phase))
    (retract (x ?t2 + ?phase))
    (asserts (xy-3 ?t1 + - ?phase)))

;; 2t pair and negative symptom in y-axis

(defrule rule-xy-24
    (xy ?t1 ?phase)
    (y ?t2 - ?phase)
```

```
    (test (= ?t2 (+ ?t1 *nteeth*)))
 =>
    (retract (xy ?t1 ?phase))
    (retract (y ?t2 - ?phase))
    (asserts (y-2-x ?t2 - ?phase)))

;; 2t pair and positive symptom in y-axis

(defrule rule-xy-25
    (xy ?t1 ?phase)
    (y ?t2 + ?phase)
    (test (= ?t2 (+ ?t1 1)))
 =>
    (retract (xy ?t1 ?phase))
    (retract (y ?t2 + ?phase))
    (asserts (xy-3 ?t1 - + ?phase)))

;; pi pair and negative symptom in y-axis

(defrule rule-y-21
    (declare (priority 10))
    (y-2 ?t1 ?shape ?phase)
    (y ?t2 - ?phase)
    (test (= ?t2 (+ ?t1 *nteeth*)))
 =>
    (retract (y-2 ?t1 ?shape ?phase))
    (retract (y ?t2 - ?phase))
    (asserts (y-3 ?t2 ?shape ?phase)))

;; pi pair and positive symptom in y-axis

(defrule rule-y-22
    (y-2 ?t1 - ?phase)
    (y ?t2 + ?phase)
    (test (= ?t2 (+ ?t1 1)))
 =>
    (retract (y-2 ?t1 - ?phase))
    (retract (y ?t2 + ?phase))
    (asserts (y-3 ?t1 + ?phase)))

;; pi pair in y-axis and negative symptom in x-axis

(defrule rule-xy-26
    (y-2 ?t1 - ?phase)
    (x ?t1 - ?phase)
 =>
```

```
    (retract (y-2 ?t1 - ?phase))
    (retract (x ?t1 - ?phase))
    (asserts (xy-3 ?t1 - - ?phase)))

;;;
;;; Third layer for symptom and second layer integration: Decision Panel
;;;

(defrule rule-x-31
    (x-3 ?t1 ?shape ?phase)
    (x ?t2 - ?phase)
    (test (and (= ?t2 (+ ?t1 *nteeth*))
               (eq *phase-y* 'idle)))
 =>
    (warn-damage ?t2)
    (retract (x-3 ?t1 ?shape ?phase))
    (retract (x ?t2 - ?phase))
    (asserts (x-3 ?t2 - ?phase)))

(defrule rule-x-31-1
    (x-3 ?t1 + ?phase)
    (x ?t2 - ?phase)
    (test (and (= ?t2 (+ ?t1 *nteeth*))
               (neq *phase-y* 'idle)))
 =>
    (retract (x-3 ?t1 + ?phase))
    (retract (x ?t2 - ?phase))
    (asserts (x-3 ?t2 perfect ?phase)))

(defrule rule-x-32
    (x-3 ?t1 ?shape ?phase)
    (x ?t2 + ?phase)
    (test (and (= ?t2 (+ ?t1 1))
               (eq *phase-y* 'idle)))
 =>
    (warn-damage ?t2)
    (retract (x ?t2 + ?phase)))

(defrule rule-x-32-1
    (x-3 ?t1 - ?phase)
    (x ?t2 + ?phase)
    (test (and (= ?t2 (+ ?t1 1))
               (neq *phase-y* 'idle)))
 =>
    (retract (x-3 ?t1 - ?phase))
    (retract (x ?t2 + ?phase))
```

```
    (asserts (x-3 ?t1 + ?phase)))

(defrule rule-x-32-2
    (x-3 ?t1 perfect ?phase)
    (x ?t2 + ?phase)
    (test (and (= ?t2 (+ ?t1 1))
               (neq *phase-y* 'idle)))
 =>
    (retract (x-3 ?t1 perfect ?phase))
    (retract (x ?t2 + ?phase))
    (asserts (x-perfect-2 ?t1 + ?phase)))

(defrule rule-x-32-3
    (x-perfect-2 ?t1 + ?phase)
    (x ?t2 - ?phase)
    (test (= ?t2 (+ ?t1 *nteeth*)))
 =>
    (retract (x-perfect-2 ?t1 + ?phase))
    (retract (x ?t2 - ?phase))
    (asserts (x-perfect-3 ?t2 - ?phase)))

(defrule rule-x-32-4
    (x-perfect-3 ?t1 - ?phase)
    (x ?t2 + ?phase)
    (test (= ?t2 (+ ?t1 1)))
 =>
    (warn-damage ?t2)
    (retract (x-perfect-3 ?t1 - ?phase))
    (retract (x ?t2 + ?phase))
    (asserts (x-perfect-3 ?t1 + ?phase)))

(defrule rule-x-32-5
    (x-perfect-3 ?t1 + ?phase)
    (x ?t2 - ?phase)
    (test (= ?t2 (+ ?t1 *nteeth*)))
 =>
    (warn-damage ?t2)
    (retract (x-perfect-3 ?t1 + ?phase))
    (retract (x ?t2 - ?phase))
    (asserts (x-perfect-3 ?t2 - ?phase)))

(defrule rule-x-33
    (x-3 ?t1 ?shape ?phase)
    (y ?t1 - ?phase)
 =>
    (warn-damage ?t1)
```

```
    (retract (x-3 ?t1 ?shape ?phase))
    (retract (y ?t1 - ?phase))
    (asserts (xy-3 ?t1 - - ?phase)))

(defrule rule-xy-31
    (xy-3 ?t1 ?shape-x ?shape-y ?phase)
    (x ?t2 - ?phase)
    (test (= ?t2 (+ ?t1 *nteeth*)))
 =>
    (warn-damage ?t2)
    (retract (xy-3 ?t1 ?shape-x ?shape-y ?phase))
    (retract (x ?t2 - ?phase))
    (asserts (x-2-y ?t2 ?shape-x ?shape-y ?phase)))

(defrule rule-xy-32
    (xy-3 ?t1 ?shape-x ?shape-y ?phase)
    (y ?t2 - ?phase)
    (test (= ?t2 (+ ?t1 *nteeth*)))
 =>
    (warn-damage ?t2)
    (retract (xy-3 ?t1 ?shape-x ?shape-y ?phase))
    (retract (y ?t2 - ?phase))
    (asserts (y-3 ?t2 ?shape-y ?phase)))

(defrule rule-xy-33
    (xy-3 ?t1 ?shape-x ?shape-y ?phase)
    (x ?t2 + ?phase)
    (test (= ?t2 (+ ?t1 1)))
 =>
    (warn-damage ?t2)
    (retract (xy-3 ?t1 ?shape-x ?shape-y ?phase))
    (retract (x ?t2 + ?phase))
    (asserts (xy-3 ?t1 + ?shape-y ?phase)))

(defrule rule-xy-34
    (xy-3 ?t1 ?shape-x ?shape-y ?phase)
    (y ?t2 + ?phase)
    (test (= ?t2 (+ ?t1 1)))
 =>
    (warn-damage ?t2)
    (retract (xy-3 ?t1 ?shape-x ?shape-y ?phase))
    (retract (y ?t2 + ?phase))
    (asserts (xy-3 ?t1 ?shape-x perfect ?phase)))

(defrule rule-xy-35
    (x-2-y ?t1 ?shape-x ?shape-y ?phase)
```

```
    (x ?t2 - ?phase)
    (test (= ?t2 (+ ?t1 *nteeth*)))
=>
    (warn-damage ?t2)
    (retract (x-2-y ?t1 ?shape-x ?shape-y ?phase))
    (retract (x ?t2 - ?phase))
    (asserts (x-3 ?t2 - ?phase)))

(defrule rule-xy-36
    (x-2-y ?t1 ?shape-x ?shape-y ?phase)
    (x ?t2 + ?phase)
    (test (= ?t2 (+ ?t1 1)))
=>
    (warn-damage ?t2)
    (retract (x-2-y ?t1 ?shape-x ?shape-y ?phase))
    (retract (x ?t2 + ?phase))
    (asserts (x-3 ?t1 + ?phase)))

(defrule rule-xy-37
    (x-2-y ?t1 ?shape-x ?shape-y ?phase)
    (y ?t1 - ?phase)
=>
    (warn-damage ?t1)
    (retract (x-2-y ?t1 ?shape-x ?shape-y ?phase))
    (retract (y ?t1 - ?phase))
    (asserts (xy-3 ?t1 ?shape-x ?shape-y ?phase)))

(defrule rule-xy-38
    (x-3 ?t1 ? ?phase)
    (y-3 ?t2 ? ?phase)
    (test (= (abs (- ?t1 ?t2)) 1))
=>
    (warn-damage (max ?t1 ?t2)))

(defrule rule-y-31
    (y-3 ?t1 ?shape ?phase)
    (y ?t2 - ?phase)
    (test (and (= ?t2 (+ ?t1 *nteeth*))
               (eq *phase-x* 'idle)))
=>
    (warn-damage ?t2)
    (retract (y-3 ?t1 ?shape ?phase))
    (retract (y ?t2 - ?phase))
    (asserts (y-3 ?t2 - ?phase)))

(defrule rule-y-31-1
```

```
    (y-3 ?t1 + ?phase)
    (y ?t2 - ?phase)
    (test (and (= ?t2 (+ ?t1 *nteeth*))
               (neq *phase-x* 'idle)))
=>
    (retract (y-3 ?t1 + ?phase))
    (retract (y ?t2 - ?phase))
    (asserts (y-3 ?t2 perfect ?phase)))

(defrule rule-y-32
    (y-3 ?t1 ?shape ?phase)
    (y ?t2 + ?phase)
    (test (and (= ?t2 (+ ?t1 1))
               (eq *phase-x* 'idle)))
=>
    (warn-damage ?t2)
    (retract (y ?t2 + ?phase)))

(defrule rule-y-32-1
    (y-3 ?t1 - ?phase)
    (y ?t2 + ?phase)
    (test (and (= ?t2 (+ ?t1 1))
               (neq *phase-x* 'idle)))
=>
    (retract (y-3 ?t1 - ?phase))
    (retract (y ?t2 + ?phase))
    (asserts (y-3 ?t1 + ?phase)))

(defrule rule-y-32-2
    (y-3 ?t1 perfect ?phase)
    (y ?t2 + ?phase)
    (test (and (= ?t2 (+ ?t1 1))
               (neq *phase-x* 'idle)))
=>
    (retract (y-3 ?t1 perfect ?phase))
    (retract (y ?t2 + ?phase))
    (asserts (y-perfect-2 ?t1 + ?phase)))

(defrule rule-y-32-3
    (y-perfect-2 ?t1 + ?phase)
    (y ?t2 - ?phase)
    (test (= ?t2 (+ ?t1 *nteeth*)))
=>
    (retract (y-perfect-2 ?t1 + ?phase))
    (retract (y ?t2 - ?phase))
    (asserts (y-perfect-3 ?t2 - ?phase)))
```

```
(defrule rule-y-32-4
    (y-perfect-3 ?t1 - ?phase)
    (y ?t2 + ?phase)
    (test (= ?t2 (+ ?t1 1)))
 =>
    (warn-damage ?t2)
    (retract (y-perfect-3 ?t1 - ?phase))
    (retract (y ?t2 + ?phase))
    (asserts (y-perfect-3 ?t1 + ?phase)))

(defrule rule-y-32-5
    (y-perfect-3 ?t1 + ?phase)
    (y ?t2 - ?phase)
    (test (= ?t2 (+ ?t1 *nteeth*)))
 =>
    (warn-damage ?t2)
    (retract (y-perfect-3 ?t1 + ?phase))
    (retract (y ?t2 - ?phase))
    (asserts (y-perfect-3 ?t2 - ?phase)))

(defrule rule-y-33
    (y-3 ?t1 ?shape ?phase)
    (x ?t1 - ?phase)
 =>
    (warn-damage ?t1)
    (retract (y-3 ?t1 ?shape ?phase))
    (retract (x ?t1 - ?phase))
    (asserts (xy-3 ?t1 - ?shape ?phase)))

;;;
;;; Clear useless hypotheses
;;;

(defrule rule-clear-1
    (declare (priority 100))
    (x ?t1 ?s ?phase)
    (x ?t2 ? ?)
    (test (> (- ?t2 ?t1) *nteeth*))
 =>
    (retract (x ?t1 ?s ?phase)))

(defrule rule-clear-2
    (declare (priority 100))
    (y ?t1 ?s ?phase)
    (y ?t2 ? ?)
```

```
    (test (> (- ?t2 ?t1) *nteeth*))
=>
    (retract (y ?t1 ?s ?phase)))

(defrule rule-clear-11
    (declare (priority 100))
    (x-2 ?t1 ?s ?phase)
    (x ?t2 ? ?)
    (test (> (- ?t2 ?t1) *nteeth*))
=>
    (retract (x-2 ?t1 ?s ?phase)))

(defrule rule-clear-12
    (declare (priority 100))
    (y-2 ?t1 ?s ?phase)
    (y ?t2 ? ?)
    (test (> (- ?t2 ?t1) *nteeth*))
=>
    (retract (y-2 ?t1 ?s ?phase)))

(defrule rule-clear-13
    (declare (priority 100))
    (x-perfect-2 ?t1 ?s ?phase)
    (x ?t2 ? ?)
    (test (> (- ?t2 ?t1) *nteeth*))
=>
    (retract (x-perfect-2 ?t1 ?s ?phase)))

(defrule rule-clear-14
    (declare (priority 100))
    (y-perfect-2 ?t1 ?s ?phase)
    (y ?t2 ? ?)
    (test (> (- ?t2 ?t1) *nteeth*))
=>
    (retract (y-perfect-2 ?t1 ?s ?phase)))


(defrule rule-clear-21
    (declare (priority 100))
    (x-3 ?t1 ?s ?phase)
    (x ?t2 ? ?)
    (test (> (- ?t2 ?t1) *nteeth*))
=>
    (retract (x-3 ?t1 ?s ?phase)))

(defrule rule-clear-22
```

```
    (declare (priority 100))
    (y-3 ?t1 ?s ?phase)
    (y ?t2 ? ?)
    (test (> (- ?t2 ?t1) *nteeth*))
 =>
    (retract (y-3 ?t1 ?s ?phase)))

(defrule rule-clear-23
    (declare (priority 100))
    (x-perfect-3 ?t1 ?s ?phase)
    (x ?t2 ? ?)
    (test (> (- ?t2 ?t1) *nteeth*))
 =>
    (retract (x-perfect-3 ?t1 ?s ?phase)))

(defrule rule-clear-24
    (declare (priority 100))
    (y-perfect-3 ?t1 ?s ?phase)
    (y ?t2 ? ?)
    (test (> (- ?t2 ?t1) *nteeth*))
 =>
    (retract (y-perfect-3 ?t1 ?s ?phase)))
```

```
;;; -*- Mode:Common-Lisp; Package:USER; Base:10 -*-
;;
;; This is an additional constraint for operation with
;; spindle speed near high natural frequency
;;

(defrule constraint-1
    (spindle-speed ?speed)
    (test (and (>= ?speed 3100) (<= ?speed 3800)))
 =>
    (send *feature-interpreter* :set-transient-threshold 60)
    (send *feature-interpreter* :set-steady-threshold 25))

;;
;; This part is to adjust feature critera, depending on
;; the belief level of sensory channels by expectation mechanism
;;

(defrule ambiguous-region-1
    (feature-interpreter ?configuration)
    (test (or (equal ?configuration 'x-ambiguous-y-ignore)
    (equal ?configuration 'x-ignore-y-ambiguous)
    (equal ?configuration 'x-ambiguous-y-ambiguous)))
 =>
    (send *feature-interpreter* :set-transient-threshold 40)
    (send *feature-interpreter* :set-steady-threshold 15))
;;
;; This rule is for changing the state threshold and the
;; ambiguity threshold, depending on the workpiece material.
;; default value: *sth* = 40 and *ambiguity-threshold* = 100
;;

(defrule state-threshold-1
    (workpiece material ?name)
    (test (equal ?name 'cast-iron))
 =>
    (SETQ *sth* 40)
    (SETQ *ambiguity-threshold* 100))

(defrule state-threshold-2
    (workpiece material ?name)
    (test (equal ?name 'aluminium))
 =>
    (SETQ *sth* 40)
    (SETQ *ambiguity-threshold* 100))
```

```
(defrule state-threshold-3
    (workpiece material ?name)
    (test (equal ?name 'steel))
=>
    (SETQ *sth* 60)
    (SETQ *ambiguity-threshold* 120))
```

REFERENCES

[Al85] Ali, M., and Scharnhorst, D. A., "Sensor-Based Fault Diagnosis in a Flight Expert System," *Proc. of the Second Conference on Artificial Intelligence Applications*, IEEE Computer Society, pp. 49-52, Washington D.C., 1985.

[Al87a] Altintas, Y., "In-Process Detection of Tool Breakage using Time Series Monitoring of Cutting Forces," submitted for publication, *International Journal of Machine Tools Manufacturing*, 1987.

[Al87b] Altintas, Y., and Yellowley, I., "In-Process Detection of Tool Failure in Milling Using Cutting Force Models," *Proc. of ASME Winter Annual Meeting*, vol. 26, pp. 1-16, 1987.

[Br85] Brownston, L., Farrell, R., Kant, E., and Martin, N., *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming*, Addison-Wesley, Reading, Massachusetts, 1985.

[Ch84] Chester, D., Lamb, D., and Dhurjati, P., "Rule- Based Computer Alarm Analysis in Chemical Process Plants," *Proc. of the Seventh Annual Conference on Computer Technology*, IEEE Computer Society, pp. 22-29, Washington D.C., 1984.

[Cl87] Clayton, B. D., *ART Programming Tutorial*, Inference Corporation, Los Angeles, California, 1987.

[DA87] D'Ambrosio, B., Fehling, M. R., Forrest, S., Raulefs, P., and Wilber, B. M., "Real-Time Process Management for Material Composition in Chemical Manufacturing," *IEEE Expert*, vol. 2, no. 2, pp. 80-93, 1987.

[Do80] Dornfeld, D., and Kannatey-Asibu, E., "Acoustic Emission During Orthogonal Metal Cutting," *International Journal of Mechanical Science*, vol. 22, pp. 285-296, 1980.

[Do87] Doyle, R. J., Sellers, S. M., and Atkinson, D. J., "Predictive Monitoring Based on Causal Simulation," *Proc. of the Second Annual Research Forum*, NASA Ames Research Center, pp. 44-59, Moffett Field , California, 1987.

[Ha86] Hamilton, M., "SCARES - A Space Control Anomaly Resolution Expert System," *Proc. of the 1986 Expert Systems in Government Conference*, IEEE Computer Society, pp. 436-443, Washington D.C., 1986.

[Is88] Ishikawa, T., and Yamada, Y., "Fracture Detection of Cutting Edge in Multipoint Tools," *Proc. of 16th North American Manufacturing Research Conference*, pp. 256-263, 1988.

[Iw77] Iwata, K., and Moriwaki, T., "An Application of Acoustic Emission to In-Process Sensing of Tool Wear," *Annals of the CIRP*, vol. 25/1, pp. 21-26, 1977.

[Ka82] Kannatey-Asibu, E., "Acoustic Emission Sensing of Tool Wear in Metal Cutting - A General Overview and Problem Areas," *Proc. of 10th North American Metal Work Research Conference*, May 1982.

[Ka81] Kannatey-Asibu, E., and Dornfeld, D., "Quantitative Relationships for Acoustic Emission from Orthogonal Metal Cutting," *Journal of Engineering for Industry*, ASME, vol. 103, pp. 330-340, 1981.

[Ko60] Koenigsberger, F., and Sabberwal, A., "Chip Section and Cutting Force during the Milling Operation," *Annals of the CIRP*, 1960.

[La88] Laffey, T. J., Cox, P. A., Schmidt, J. L., Kao, S. M., and Read, J. Y., "Real-Time Knowledge-Based Systems," *AI Magazine*, pp. 27-45, Spring 1988.

[Le88] Lee, M., Wildes, D., Hayashi S., and Keramati B., "Effects of Tool Geometry on Acoustic Emission Intensity," *Annals of the CIRP*, vol. 37/1, pp. 57- 60, 1988.

[Li87] Liang, S., and Dornfeld, D., "Detection of Cutting Tool Wear using Adaptive Time Series Modelling of Acoustic Emission Signal," *Proc. of ASME Winter Annual Meeting*, vol. 26, pp. 27-38, 1987.

[Ma41] Martelotti, M., "An Analysis of the Milling Process," *Trans. of ASME*, vol. 63, pp. 677-700, 1941.

[Ma45] Martelotti, M., "An Analysis of the Milling Process -- Down Milling," *Trans. of ASME*, vol. 67, pp. 233-251, 1945.

[Ma88] Mannering, D., *Tool Kit Specification for the Explorer and TMS32020 Odyssey Board*, preliminary draft, Speech and Image Understanding Lab., TI Computer Science Center, Texas Instruments Inc., Dallas, Texas, 1988.

[Ma82] Matushima, K., Bertok, P., and Sata, T., "In-Process Detection of Tool Breakage by Monitoring the Spindle Motor Current of a Machine Tool," *Measurement and Control for Batch Manufacturing, Winter Annual Meeting*, ASME, pp. 145-153, 1982.

[Mc87] McTamaney, L. S., "Real-Time Intelligence Control," *IEEE Expert*, pp. 55-68, Winter 1987.

[Mo80] Moriwaki, T., "Detection of Tool Fracture by Acoustic Emission Measurement," *Annals of the CIRP*, vol. 29/1, pp. 35-40, 1980.

[Ra75] Rabiner, L. R., Sambur, M. R., and Schmidt, C. E., "Applications of Nonlinear Smoothing Algorithm to Speech Processing," *IEEE Trans. on ASSP*, vol. 23, no. 6, pp. 552-557, 1975.

[Ra87] Rangwala, S., and Dornfeld, D., "Integration of Sensors via Neural Networks for Detection of Tool Wear States," *Intelligent and Integrated Manufacturing Analysis and Synthesis*, edited by C. R. Liu and et al., pp. 109-120, presented at the Winter Annual Meeting of the ASME, Dec. 1987.

[Sc85] Scarl, E., Jamieson, J., and Delaune, C., "A Fault Detection and Isolation Method Applied to Liquid Oxygen Loading for the Space Shuttle," *Proc. of IJCAI-85*, pp. 414-416, Los Angeles, California, Aug. 1985.

[St86] Stefik, M., and Bobrow, D., "Object oriented programming: themes and variations," *AI Magazine*, vol. 6, no. 4, pp. 40-62, 1986.

[Ta86] Takata, S., Ahn, J. H., Miki, M., Miyao, Y., and Sata, T., "A Sound Monitoring System for Fault Detection of Machines and Machineing States," *Annals of the CIRP*, vol. 35/1, pp. 289-292, 1986.

[Ta87] Tarn, M., and Tomizuka, M., "On-line Monitoring of Tool and Cutting Conditions in Milling," *Proc. of ASME Winter Annual Meeting*, vol. 26, pp. 17-25, 1987.

[Ta88] Tarng, Y., "Use of Various Signals for Milling Cutter Breakage Detection," *Ph.D. Dissertation, University of Florida*, 1988.

[Tl83] Tlusty, J., and Andrews, G. C., "A Critical Review of Sensors for Unmanned Machining," *Annals of the CIRP*, vol. 32/2, pp. 563-572, 1983.

[Tl88] Tlusty, J., and Tarng, Y., "Sensing Cutter Breakage in Milling," *Annals of the CIRP*, vol. 37/1, pp. 45-51, 1988.

[Wa86] Waterman, D. A., *A Guide to Expert Systems*, Addison-Wesley, Reading, Massachusetts, 1986.

BIOGRAPHICAL SKETCH

Taehwan Yoon was born January 2, 1953, in Pusan, Korea. He received his bachelor's degree in electronic engineering, graduating with honors (magna cum laude), from the Seoul National University in Febrary, 1975, and his master's degree in electrical engineering from the Korea Advanced Institute of Science and Technology in Febrary, 1977.

From 1977 to 1984, he had been with the Agency for Defense Development (ADD), Daejeon, Korea, where he was a senior researcher. At the Agency for Defense Development, he was awarded the Defense Development Silver Prize in recognition of R & D achievement in January, 1984.

In September, 1984, he and his family came to Gainesville to do his Ph.D. work with the Department of Electrical Engineering, the University of Florida. Since 1987, he has worked in the area of digital signal processing and computer application at the University of Florida.

I certify that I have read this study and that in my opinion it confirms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Jose C. Principe, Chair
Associate Professor
of Electrical Engineering

I certify that I have read this study and that in my opinion it confirms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Jack R. Smith, Cochair
Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it confirms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Shamkant Navathe
Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it confirms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

A. Antonio Arroyo
Associate Professor
of Electrical Engineering

I certify that I have read this study and that in my opinion it confirms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Jiri Tlusty
Graduate Research Professor
of Mechanical Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

May 1990

for Hubert A. Bevis
Winfred M. Phillips
Dean, College of Engineering

Madelyn M. Lockhart
Dean, Graduate School